APPROXIMATION ALGORITHMS FOR INTEGRAL CONCURRENT FLOW,
INDEPENDENT SET OF RECTANGLES, AND FIRE CONTAINMENT PROBLEMS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
PARINYA CHALERMSOOK

CHICAGO, ILLINOIS
JUNE 2012
# TABLE OF CONTENTS

LIST OF FIGURES .............................................. vi
LIST OF TABLES ............................................. vii
ACKNOWLEDGMENTS ........................................ viii
ABSTRACT ................................................ x
INTRODUCTION ............................................. 1

I INTEGRAL CONCURRENT FLOW

1 PRELIMINARIES ........................................... 14
  1.1 Demands and Routing ................................... 14
  1.2 Sparsest Cut and the Flow-Cut Gap .................... 15
  1.3 Well-linkedness ...................................... 16
    1.3.1 Non-constructive decomposition .................. 18
    1.3.2 Constructive Version ............................ 24
    1.3.3 The Grouping Technique ......................... 27
  1.4 Bandwidth Property and Critical Clusters ........... 28
    1.4.1 Proof of Lemma 4 ................................ 29
    1.4.2 Proof of Lemma 5 ................................ 31
  1.5 Routing across Small and Critical Clusters ........ 35
    1.5.1 Proof of Theorem 9 .............................. 36
    1.5.2 Routing across Critical Clusters ................ 37
  1.6 Organization and List of Parameters ................ 43

2 A GRAPH SPLITTING THEOREM ............................. 46
  2.1 Graph Decomposition ................................ 46
  2.2 Proof of Theorem 15 ................................ 50
    2.2.1 Joined Tendril Graph ........................... 50
    2.2.2 Graph $H$ ......................................... 53

3 BASIC INTEGRAL CONCURRENT FLOW ....................... 60
  3.1 The Algorithm ....................................... 60
  3.2 Proof of Theorem 21 ................................ 64
  3.3 Proof of Theorem 22 ................................ 67
  3.4 Hardness of basic-ICF ............................... 72

4 GROUP INTEGRAL CONCURRENT FLOW ......................... 74
  4.1 Hardness of group-ICF ............................... 74
    4.1.1 Max-Min Interval Scheduling ..................... 74
    4.1.2 Hardness of MMIS ............................... 76
  4.2 An Algorithm for group-ICF .......................... 81
4.2.1 Split Instances and Good $Q$-$J$ Decompositions ........................................ 82
4.2.2 The Algorithm ................................................................................................ 95
4.2.3 Proof of Theorem 31 ..................................................................................... 102

5 EXTENSIONS, OTHER RESULTS AND CONCLUDING REMARKS .................. 104
5.1 Arbitrary Edge Capacities and Demands ....................................................... 104
5.2 Gaps Between Fractional and Integral group-ICF ........................................... 106
  5.2.1 Lower Bound .............................................................................................. 106
  5.2.2 Upper Bound .............................................................................................. 109
5.3 Relationship between Problems ..................................................................... 110
5.4 Conclusion and Open Problems ..................................................................... 110

II MAXIMUM INDEPENDENT SET OF RECTANGLES

6 PRELIMINARIES ............................................................................................... 113
  6.1 Preliminaries .................................................................................................... 113
  6.2 Maximum Independent Set via Coloring ...................................................... 115
  6.3 Our Results ..................................................................................................... 119
  6.4 Organization .................................................................................................... 120

7 COLORING ALGORITHMS ............................................................................... 121
  7.1 Sparse Instances ............................................................................................. 121
  7.2 Instances that are not sparse .......................................................................... 122
  7.3 $O(q^2)$-Coloring Algorithm ........................................................................... 124
  7.4 Structure Lemma ............................................................................................. 125
  7.5 An $O(q^{3/2})$ Coloring for Restricted Setting .............................................. 128
  7.6 An $O(q \log q)$ Coloring ................................................................................. 129

8 EXTENSIONS AND CONCLUDING REMARKS ON MISR ......................... 135
  8.1 MISR in Higher Dimension ........................................................................... 135
  8.2 Applications and Connections to Other Problems ....................................... 136
  8.3 Integrality Gap of LP ..................................................................................... 137
  8.4 Dealing with Containment Intersection ....................................................... 140
  8.5 A Short Survey on Rectangle Coloring ......................................................... 143
  8.6 Conclusion and Open problems ..................................................................... 144

III RESOURCE MINIMIZATION FOR FIRE CONTAINMENT

9 PRELIMINARIES ............................................................................................... 148
  9.1 Formal Problem Definition ............................................................................ 148
  9.2 LP Relaxation .................................................................................................. 148
  9.3 Simplified LP for Special Cases .................................................................... 149
  9.4 Our Results ..................................................................................................... 154
  9.5 Organization .................................................................................................... 155
10 APPROXIMATION ALGORITHMS ........................................ 156
  10.1 Trees .............................................................. 156
    10.1.1 High-Level Overview .................................... 156
    10.1.2 The Algorithm ........................................... 158
    10.1.3 Proof of Theorem 39 .................................... 161
    10.1.4 Proof of Theorem 41 .................................... 168
  10.2 Directed Layered Graphs ..................................... 175
  10.3 General Graphs ................................................. 176

11 LOWER BOUNDS ON THE LP INTEGRALITY GAP ................. 180
  11.1 Trees .............................................................. 180
    11.1.1 Stronger LP Relaxation ................................ 180
    11.1.2 Construction .............................................. 182
    11.1.3 Analysis .................................................. 184
  11.2 Directed Layered Graphs ..................................... 188
  11.3 General Graphs ................................................. 188

12 CONCLUSION AND OPEN PROBLEMS ............................. 191

REFERENCES ............................................................ 193
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Illustration for Lemma 1</td>
<td>19</td>
</tr>
<tr>
<td>1.2</td>
<td>Illustration for Claim 1</td>
<td>20</td>
</tr>
<tr>
<td>1.3</td>
<td>Illustration for Lemma 2</td>
<td>22</td>
</tr>
<tr>
<td>5.1</td>
<td>The gap example construction</td>
<td>107</td>
</tr>
<tr>
<td>5.2</td>
<td>The connections between problems related to ICF. The arrows indicate that one problem is a special case of another, and the line with ? indicates an unknown (but seemingly existing) relationship.</td>
<td>110</td>
</tr>
<tr>
<td>6.1</td>
<td>Three possible intersection types</td>
<td>113</td>
</tr>
<tr>
<td>7.1</td>
<td>An instance that is not sparse</td>
<td>123</td>
</tr>
<tr>
<td>7.2</td>
<td>Proof of Lemma 7.2 when $x' &gt; x$. The black stripe shows the intersection region of $C'$</td>
<td>126</td>
</tr>
<tr>
<td>7.3</td>
<td>The description of iteration $i$ of the algorithm</td>
<td>131</td>
</tr>
<tr>
<td>7.4</td>
<td>Rectangles $Q \in X''$ and $Q' \in X'$.</td>
<td>132</td>
</tr>
<tr>
<td>7.5</td>
<td>The rectangles in $Q_R$. The four intersection points between rectangles in $Q_R$ and $R$ denote the representative points of $R$.</td>
<td>134</td>
</tr>
<tr>
<td>8.1</td>
<td>Schematic View of $I_j$</td>
<td>138</td>
</tr>
<tr>
<td>8.2</td>
<td>Instance $I_1$</td>
<td>139</td>
</tr>
<tr>
<td>8.3</td>
<td>Instance $I_{j+1}$</td>
<td>139</td>
</tr>
<tr>
<td>9.1</td>
<td>An example that not choosing any vertex from layer 1 can give advantage. Note that the triangles are used to denote terminals.</td>
<td>153</td>
</tr>
<tr>
<td>11.1</td>
<td>Type-1 spider of width $w$ and height $h$</td>
<td>182</td>
</tr>
<tr>
<td>11.2</td>
<td>Type-2 spider of width $w$ and height $h$</td>
<td>182</td>
</tr>
</tbody>
</table>
LIST OF TABLES

9.1 Summary of Our Results ........................................ 154
ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my advisors Julia Chuzhoy and Janos Simon. I started working with Julia in my third year of Ph.D, when I was in a bad situation and was on the verge of quitting the program. Julia got me back on research track by giving me interesting problems, sage advises, and continuing encouragement. Indeed, I would probably have ended up as a banker right now if I did not meet with Julia. Thanks so much, Julia, for getting me back to do something I really like. Besides technical stuffs, she also taught me about presentation and writing. I feel very lucky to have Julia as my advisor.

Janos has always been very encouraging and understanding. He always has excellent advises about everything ranging from academic aspects to life in general. He is always available for discussion and willing to listen to everything about research. He is also very nice and patient. Almost everyone who knows me knows that I have a bad habit of doing things in the last minute, and Janos is among the persons who suffer the most from this habit. I gave him a short notice about almost every important matter, but he has been surprisingly patient.

I would like to thank Sasha Razborov for agreeing to be on my committee both in my candidacy exam and the thesis defense, and Lance Fortnow for advising me in the early years of my Ph.D. I especially thank Lance for teaching me a lot about complexity theory during my first two years. I took four classes with Lance, which turned out to be very useful even when I read algorithms’ papers these days. I also thank Laci Babai for showing me how to make a basic mathematics course interesting. Indeed, his discrete mathematics course is very enjoyable even for those who already knew the material.

I am grateful to both University of Chicago and TTI-C for providing me office space and funding in the past seven years. I also thank Julia’s NSF CAREER grant CCF-0844872 that has been the main source of my funding in the past couple of years.

Friends are also a great source of advises and joy. In particular, Bundit, Danupon and Jittat have been both friends and research collaborators. I thank Jittat for introducing me
to the fascinating area of algorithms and for being my regular drinking partner when I was in Thailand. He also taught me how to distinguish between good and bad alcohols. Bundit and Danupon are always available on Skype whenever I need to discuss something, especially silly little things that I find too shameful to ask others. I thank them for a countless number of hours they gave me. I would like thank all my friends at the University of Chicago for making my Ph.D years enjoyable.

Finally, almost every achievement in my life would not have been possible without my family. I thank my wife and my parents for their unconditional love and support. Due to a series of unfortunate situations, my wife and I have rarely been able to be together, but her patience and understanding made this possible. Now I do believe that love, patience, and understanding make a long distant relationship possible.

My parents have always been supportive through my whole life. They raised me to be the person I am now and offered support whenever I needed. My brother has spent a lot of time with me in my childhood. I thank him for a joyful conversation whenever we got a chance to meet.
ABSTRACT

We study the approximability of three NP-hard combinatorial optimization problems arising in various areas of computer science: Integral Concurrent Flow, Maximum Independent Set of Rectangles, and Fire Containment problems.

In the first part, we study the integral counterpart of the classical Maximum Concurrent Flow problem, as well as its variants. In the basic version of this problem (basicICF), we are given an undirected n-vertex graph $G$ with edge capacities $c(e)$, a subset $T$ of vertices called terminals, and a demand $D(t, t')$ for every pair $(t, t')$ of the terminals. The goal is to find the maximum value $\lambda$, and a collection $P$ of paths, such that every pair $(t, t')$ of terminals is connected by $\lfloor \lambda D(t, t') \rfloor$ paths in $P$, and the number of paths containing any edge $e$ is at most $c(e)$. We show a poly log $n$ approximation algorithm that violates the edge capacities by only a constant factor. The key ingredient of our results is a new graph splitting technique which could be of independent interest. The results in this part are based on joint work with Julia Chuzhoy, Alina Ene, and Shi Li [29] in STOC 2012.

In the second part of the thesis, we turn to a geometric problem. Given a collection of rectangles, our goal is to select a non-overlapping sub-collection of maximum cardinality. This problem has many applications, as well as connections to other problems in various areas of computer science. The problem was introduced in 1997 together with a simple $O(\log n)$ approximation algorithm for it. We show an $O(\log \log n)$ approximation algorithm. Our algorithm illustrates a connection between this problem and a half-century old rectangle coloring problem. The main result in this part is from a joint work with Julia Chuzhoy [26] that appeared in SODA 2009. The presentation of the main algorithm follows the framework in my APPROX paper [25].

Finally, we focus on a resource minimization version of the fire containment problem in a standard mathematical model that is used to deal with spreading processes of fire and viruses. We show LP-rounding approximation algorithms for various classes of graphs and provide essentially tight lower bounds on the integrality gap of natural LP relaxations for these
graph classes. The main result of this part is an $O(\log^* n)$ approximation algorithm when the underlying input graph is a tree. Most results in this part appeared in SODA 2010 [27] in a joint work with Julia Chuzhoy. Some results are from an unpublished manuscript, joint with Julia Chuzhoy and Paolo Codenotti [28].
INTRODUCTION

In this thesis, we study three NP-hard combinatorial optimization problems arising in various areas of computer science: the integral concurrent flow problem, the maximum independent set of rectangles problem, and the resource minimization variant of fire containment problem. The integral concurrent flow problem belongs to the area of network flow theory. The independent set of rectangles problem is a fundamental problem in the area of computational geometry, while the fire containment problem concerns the spreading processes of fire and viruses.

Integral Concurrent Flow

Multicommodity flows are ubiquitous in computer science, and they are among the most basic and extensively studied combinatorial objects. Given an undirected $n$-vertex graph $G = (V, E)$ with capacities $c(e) > 0$ on edges $e \in E$, and a collection $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs, two standard objective functions for multicommodity flows are: Maximum Multicommodity Flow, where the goal is to maximize the total amount of flow routed between the source-sink pairs, and Maximum Concurrent Flow, where the goal is to maximize a value $\lambda$, such that $\lambda$ flow units can be simultaneously sent between every pair $(s_i, t_i)$.

Many applications require however that the routing of the demand pairs is integral, that is, the amount of flow sent on each flow-path is integral. The integral counterpart of Maximum Multicommodity Flow is the Edge Disjoint Paths problem (EDP), where the goal is to find a maximum-cardinality collection $P$ of paths connecting the source-sink pairs with no congestion. It is a standard practice to define the EDP problem on graphs with unit edge capacities, so a congestion of any solution $P$ is the maximum number of paths in $P$ sharing an edge. EDP is a classical routing problem that has been studied extensively. Robertson and Seymour [82] have shown an efficient algorithm for EDP when the number $k$ of the demand pairs is bounded by a constant, but the problem is NP-hard for general values of $k$ [66]. The
best currently known approximation algorithm, due to Chekuri, Khanna and Shepherd [37], achieves an $O(\sqrt{n})$-approximation. The problem is also known to be $\Omega(\log^{1/2-\epsilon} n)$-hard to approximate for any constant $\epsilon$, unless $\text{NP} \subseteq \text{ZTIME}(n^{\text{poly log } n})$ [7, 6]. A standard technique for designing approximation algorithms for routing problems is to first compute a multi-commodity flow relaxation of the problem, where instead of connecting the demand pairs with integral paths, we are only required to send flow between them. Such a fractional solution can usually be computed using linear programming, and it is then rounded to obtain an integral solution to the routing problem. For the EDP problem, the corresponding flow relaxation is the Maximum Multicommodity Flow problem. However, the ratio of the Maximum Multicommodity Flow solution value to the EDP solution value can be as large as $\Omega(\sqrt{n})$ in some graphs [37]. Interestingly, when the value of the global minimum cut in $G$ is $\Omega(\log^5 n)$, Rao and Zhou [79] have shown a poly log $n$-approximation algorithm for EDP, by rounding the multicommodity flow relaxation.

Much better results are known if we slightly relax the problem requirements by allowing a small congestion. Andrews [5] has shown an efficient randomized algorithm that w.h.p. routes $\Omega(\text{OPT}/\text{poly log } n)$ of the demand pairs with congestion poly($\log \log n$), where $\text{OPT}$ is the value of the optimal solution with no congestion for the given EDP instance, and Chuzhoy [40] has shown an efficient randomized algorithm that w.h.p. routes $\Omega(\text{OPT}/\text{poly log } k)$ of the demand pairs with a constant congestion. In fact the number of demand pairs routed by the latter algorithm is within a poly log $k$-factor of the Maximum Multicommodity Flow value.

Assume now that we are given an instance where every demand pair $(s_i, t_i)$ can simultaneously send $D$ flow units to each other with no congestion. The algorithm of [40] will then produce a collection $P$ of $\Omega(Dk/\text{poly log } k)$ paths connecting the demand pairs, but it is possible that some pairs are connected by many paths, while some pairs have no paths connecting them. In some applications however, it is important to ensure that every demand pair is connected by many paths.
In this paper, we propose to study an integral counterpart of Maximum Concurrent Flow, called Integral Concurrent Flow (ICF). We study two versions of ICF. In the simpler basic version (basic-ICF), we are given an undirected \( n \)-vertex graph \( G = (V, E) \) with non-negative capacities \( c(e) \) on edges \( e \in E \), a subset \( T \subseteq V \) of \( k \) vertices called terminals, and a set \( D \) of demands over the terminals, where for each pair \((t_i, t_j) \in T\), a demand \( D(t_i, t_j) \) is specified. The goal is to find a maximum value \( \lambda \), and a collection \( P \) of paths, such that for each pair \((t_i, t_j) \) of terminals, set \( P \) contains at least \( \lfloor \lambda \cdot D(t_i, t_j) \rfloor \) paths connecting \( t_i \) to \( t_j \), and for each edge \( e \in E \), at most \( c(e) \) paths in \( P \) contain \( e \).

The second and the more general version of the ICF problem that we consider is the group version (group-ICF), in which we are given an undirected \( n \)-vertex graph \( G = (V, E) \) with edge capacities \( c(e) > 0 \), and \( k \) pairs of vertex subsets ((\( S_1, T_1 \)), \ldots, (\( S_k, T_k \))). For each pair \((S_i, T_i)\), we are also given a demand \( D_i \). The goal is to find a maximum value \( \lambda \), and a collection \( P \) of paths, such that for each \( 1 \leq i \leq k \), there are at least \( \lfloor \lambda \cdot D_i \rfloor \) paths connecting the vertices of \( S_i \) to the vertices of \( T_i \) in \( P \), and every edge \( e \in E \) belongs to at most \( c(e) \) paths. It is easy to see that group-ICF generalizes both the basic-ICF and the EDP problems\(^1\). As in the EDP problem, we will sometimes relax the capacity constraints, and will instead only require that the maximum edge congestion - the ratio of the number of paths containing the edge to its capacity - is bounded. We say that a set \( P \) of paths is a solution of value \( \lambda \) and congestion \( \eta \), iff for every \( 1 \leq i \leq k \), at least \( \lfloor \lambda \cdot D_i \rfloor \) paths connect the vertices of \( S_i \) to the vertices of \( T_i \), and every edge \( e \in E \) participates in at most \( \eta \cdot c(e) \) paths in \( P \). Throughout the paper, we denote by \( \lambda^* \) the value of the optimal solution to the ICF instance, when no congestion is allowed. We say that a solution \( P \) is an \( \alpha \)-approximation with congestion \( \eta \) iff for each \( 1 \leq i \leq k \), at least \( \lfloor \lambda^* \cdot D_i / \alpha \rfloor \) paths connect the vertices of \( S_i \) to the vertices of \( T_i \), and the congestion due to paths in \( P \) is at most \( \eta \).

Given a multicommodity flow \( F \), we say that it is a fractional solution of value \( \lambda \) to the

---

\(^1\) To reduce EDP to group-ICF, make \( D \) disjoint copies of the EDP instance. For each \( 1 \leq i \leq k \), let \( S_i \) contain all copies of \( s_i \) and \( T_i \) contain all copies of \( t_i \). If we can find \( AD \) paths for every group \((S_i, T_i)\), then some copy of the EDP instance will contain a solution of value at least \( \lambda k \).
group-ICF instance, iff for each demand pair \((S_i, T_i)\), at least \(\lambda \cdot D_i\) flow units are sent from the vertices of \(S_i\) to the vertices of \(T_i\), and the total amount of flow sent via any edge \(e\) is at most \(c_e\). Throughout the paper, we denote by \(\lambda_{\text{opt}}\) the value of the optimal fractional solution to the ICF problem instance. The value \(\lambda_{\text{opt}}\) can be efficiently computed by solving an appropriate LP-relaxation of the problem, for both basic-ICF and group-ICF. Observe that for basic-ICF, this is equivalent to solving the Maximum Concurrent Flow problem.

In addition to designing approximation algorithms for the ICF problem, an interesting question is the relationship between the optimal fractional and the optimal integral solutions for ICF. For example, suppose we are given a multicommodity flow, where for each \(1 \leq i \leq k\), the vertices of \(S_i\) send \(D\) flow units to the vertices of \(T_i\) simultaneously with no congestion. What is the maximum value \(\lambda\), for which we can find an integral solution where for each pair \((S_i, T_i)\) at least \(\lfloor \lambda D \rfloor\) paths connect the vertices of \(S_i\) to the vertices of \(T_i\)?

**Our Results**

Our first result is an approximation algorithm for the basic-ICF problem.

**Theorem 1** There is an efficient randomized algorithm, that, given any instance of basic-ICF, w.h.p. produces an integral solution of value \(\lambda_{\text{opt}}/\text{poly log } n\) and constant congestion, where \(\lambda_{\text{opt}}\) is the value of the optimal fractional solution with no congestion to the ICF instance.

The main technical tool that our algorithm uses is a graph decomposition similar to the one proposed by Andrews [5]. Assume first that the value of the minimum cut in graph \(G\) is polylogarithmic. We can then define \(\text{poly log } n\) new graphs \(G_1, \ldots, G_r\), where for each \(1 \leq j \leq r\), \(V(G_j) = V(G)\), and the edges in graphs \(G_j\) form a partition of the edges in \(G\). If the value of the minimum cut in \(G\) is large enough, we can furthermore ensure that the value of the minimum cut in each resulting graph \(G_j\) is \(\Omega(\text{poly log } n)\). We can then use the algorithm of Rao and Zhou [79] to find \(\lambda^* \cdot \sum_i D_i / \text{poly log } n\) paths connecting the source-sink pairs in each graph \(G_j\) separately. By appropriately choosing the subsets of source-sink pairs for
each graph $G_j$ to connect, we can obtain a polylogarithmic approximation for the basic-ICF problem instance.

Unfortunately, it is possible that the global minimum cut in graph $G$ is small. Andrews [5] in his paper on the EDP problem, suggested to get around this difficulty as follows. Let $L = \text{poly log } n$ be a parameter. For any subset $C$ of vertices in $G$, let $\text{out}(C) = E(C, V \setminus C)$ be the set of edges connecting the vertices of $C$ to the vertices of $V \setminus C$. We say that a subset $C$ of vertices is a large cluster iff $|\text{out}(C)| \geq L$, and otherwise we say that it is a small cluster. Informally, we say that cluster $C$ has the bandwidth property iff we can send $1/|\text{out}(C)|$ flow units between every pair of edges in $\text{out}(C)$ with small congestion inside the cluster $C$. Finally, we say that $C$ is a critical cluster iff it is a large cluster, and we are given a partition $\pi(C)$ of its vertices into small clusters, such that on the one hand, each cluster in $\pi(C)$ has the bandwidth property, and on the other hand, the graph obtained from $G[C]$ by contracting every cluster in $\pi(C)$ is an expander. The key observation is that if $C$ is a critical cluster, then we can integrally route demands on the edges of $\text{out}(C)$ inside $C$, by using standard algorithms for routing on expanders. The idea of Andrews is that we can use the critical clusters to “hide” the small cuts in graph $G$.

More specifically, the graph decomposition procedure of Andrews consists of two steps. In the first step, he constructs what we call a $Q$-$J$ decomposition $(Q, J)$ of graph $G$. Here, $Q$ is a collection of disjoint critical clusters and $J$ is a collection of disjoint small clusters that have the bandwidth property, and $Q \cup J$ is a partition of $V(G)$. This partition is computed in a way that ensures that every cut separating any pair of clusters in $Q$ is large, containing at least poly log $n$ edges, and moreover we can connect all edges in $\bigcup_{C \in J} \text{out}(C)$ to the edges of $\bigcup_{C \in Q} \text{out}(C)$ by paths that together only cause a small congestion.

Given a $Q$-$J$ decomposition $(Q, J)$, Andrews then constructs a new graph $H$, whose vertices are $\{v_Q \mid Q \in Q\}$, and every edge $e = (v_Q, v_{Q'})$ in $H$ is mapped to a path $P_e$ in $G$ connecting some vertex of $Q$ to some vertex of $Q'$, such that the total congestion caused by the set $\{P_e \mid e \in E\}$ of paths in graph $G$ is small. Moreover, graph $H$ preserves, to within a
polylogarithmic factor, all cuts separating the clusters of $Q$ in graph $G$. In particular, the size of the global minimum cut in $H$ is large, and any integral routing in graph $H$ can be transformed into an integral routing in $G$. This reduces the original problem to the problem of routing in the new graph $H$. Since the size of the minimum cut in graph $H$ is large, we can now apply the algorithm proposed above to graph $H$.

We revisit the $Q$-$J$ decomposition and the construction of the graph $H$ from [5], and obtain an improved construction with better parameters. In particular, it allows us to reduce the routing congestion to constant, and to reduce the powers of the logarithms in the construction parameters. The $Q$-$J$ decomposition procedure of [5] uses the tree decomposition of Räcke [77] as a black box. We instead perform the decomposition directly, thus improving some of its parameters. We also design a new well-linked decomposition procedure that may be of independent interest.

Our next result shows that basic-ICF is hard to approximate, using a simple reduction from the Congestion Minimization problem.

**Theorem 2** Given an $n$-vertex graph $G$ with unit edge capacities, a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs, and integers $c, D$, such that $Dc \leq O(\frac{\log \log n}{\log \log \log n})$, unless $\text{NP} \subseteq \text{ZPTIME}(n^{\text{poly} \log n})$, no efficient algorithm can distinguish between the following two cases: (i) There is a collection $\mathcal{P}$ of paths that causes congestion $1$, with $D$ paths connecting $s_i$ to $t_i$ for each $1 \leq i \leq k$; and (ii) Any collection $\mathcal{P}$ of paths, containing, for each $1 \leq i \leq k$, at least one path connecting $s_i$ to $t_i$, causes congestion at least $c$.

We then turn to the group-ICF problem, and prove that it is hard to approximate in the following theorem.

**Theorem 3** Suppose we are given an $n$-vertex graph $G = (V, E)$ with unit edge capacities, and a collection of pairs of vertex subsets $(S_1, T_1), \ldots, (S_k, T_k)$. Let $c$ be any integer, $0 < c \leq O(\log \log n)$ and let $D = O\left(n^{1/2)^{2c+3}}\right)$. Then unless $\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)})$, no efficient algorithm can distinguish between the following two cases: (i) There is a collection
\(P^*\) of paths that causes congestion 1, and contains, for every \(1 \leq i \leq k\), \(D\) paths connecting
the vertices of \(S_i\) to the vertices of \(T_i\); and (ii) Any set \(P^*\) of paths, containing, for each
\(1 \leq i \leq k\), at least one path connecting a vertex of \(S_i\) to a vertex of \(T_i\), causes congestion at
least \(c\).

The proof of Theorem 3 establishes a connection between group-ICF and the Machine
Minimization Scheduling problem, and then follows the hardness of approximation proof
of [41] for the scheduling problem. Finally, we show an approximation algorithm for the
group-ICF problem.

**Theorem 4** Suppose we are given an instance of group-ICF, and let \(D = \min_i \{\lambda_{OPT} \cdot D_i\}\)
be the minimum amount of flow sent between any pair \((S_i, T_i)\) in the optimal fractional
solution. Assume further that \(D \geq \Delta'\), where \(\Delta' = k \cdot \text{poly} \log n\) is a parameter whose value
we set later. Then there is an efficient randomized algorithm that finds a solution of value
\(\lambda_{OPT}/\text{poly} \log n\) with constant congestion for the group-ICF instance.

**Maximum Independent Set of Rectangles**

In Maximum Independent Set of Rectangles (MISR), we are given a collection \(\mathcal{R}\) of \(n\)-axis
parallel rectangles where each rectangle \(R \in \mathcal{R}\) is associated with weight \(w_R\). The goal of
the problem is to find a subset \(\mathcal{R}' \subseteq \mathcal{R}\) such that rectangles in \(\mathcal{R}'\) are non-overlapping,
while maximizing the total weight in \(\mathcal{R}'\). A particularly interesting special case is when all
rectangles have the same weight. We refer to this special case as unweighted MISR, which will
be our main focus. In fact, when MISR was originally introduced by Agarwal, van Kreveld,
and Suri [3], the problem was formulated as an unweighted problem, and later researchers
have found more applications by considering the more general weighted version.

It is easy to see that MISR is a special case of Maximum Independent Set problem: Given
a collection \(\mathcal{R}\), define a graph \(G = (V, E)\) where the vertices correspond to rectangles, and
there is an edge connecting two vertices if and only if two corresponding rectangles overlap.
The graph $G$ created this way is called **intersection graph** of $\mathcal{R}$. It is clear that **MISR** is equivalent to the problem of finding a maximum independent set on intersection graphs defined by axis-parallel rectangles.

**Maximum Independent Set** is arguably one of the most fundamental combinatorial optimization problems (see, e.g., [22, 58, 48]). It is known to be $n^{1-\epsilon}$ hard to approximate unless $\text{NP} = \text{ZPP}$, thus most probably ruling out the possibility of good approximation algorithms.

However, for many applications, it is enough to solve the **Maximum Independent Set** problem on restricted class of instances, which often turn out to be more tractable. For example, **Maximum Independent Set** is known to be solvable in polynomial time on interval graphs (intersection graphs of intervals on the real line). **MISR** can be seen as a generalization of this problem to two dimensions.

**MISR** is a fundamental problem in computational geometry and has come up a lot in various areas of computer science, e.g. in map labeling [3, 46, 45], network routing [21], channel admission control [72, 1, 80], and pricing [39]. Therefore, it is not surprising that **MISR** attracted a considerable amount of attention from various research communities. Since the problem is **NP**-hard [51], the main focus has been on designing approximation algorithms. Several research groups have suggested $O(\log n)$ approximation algorithms for **MISR** (see, e.g.,[76, 3].) Moreover, Berman et. al. [18] showed that a constant in the Big-oh notation can be made arbitrarily small, i.e. they show $\lceil \log k \cdot n \rceil$ approximation algorithm running in time $n^k \cdot \text{OPT}$, and the running time is improved by Chan [31] to $\min \left\{ O(n \log n + nqk^{-2}), n^{O(k/\log k)} \right\}$. Let $q$ be the maximum number of rectangles containing the same point in the plane. Lewin-Eytan, Naor, and Orda [72] designed a factor of $4q$ approximation algorithm, and (independently) Agarwal and Mustafa [2] showed an $O(q)$ approximation algorithm. We note that these results are not comparable to the results by Berman et. al. and Chan. Recently, Chan and Har-Peled [32] designed an $O(\frac{\log n}{\log \log n})$ approximation algorithm for **MISR**, and this remained the best known result.

For special cases of **MISR** where the shapes of input rectangles are restricted, better
results can be obtained. When the input rectangles are squares, unit height rectangles, or \textit{fat} rectangles, the problem is known to admit a polynomial time approximation scheme (PTAS) [30, 60, 47].

We also consider the MISR problem in higher dimensions. That is for \(d \geq 1\), a \(d\)-box graph is an intersection graph of \(d\)-dimensional axis-parallel hyper-rectangles (or boxes). We define the problem \(d\)-MISR as the \textbf{Maximum Independent Set} problem in \(d\)-box graphs. When \(d = 2\), this problem is equivalent to MISR, and for \(d = 3\) the problem is known to be APX-hard [38].

\textit{Our Results}

Our main result is an \(O(\log \log n)\) approximation algorithm for unweighted MISR. This result can be extended, by using standard techniques, to give an \(O(\log^{d-2} n \log \log n)\) approximation algorithm for unweighted \(d\)-MISR, improving upon an \(O(\log^{d-1} n)\) factor. Our algorithm is based on a rounding of a natural LP relaxation of the problem, so it implies an upper bound on the integrality gap of the LP.

\textbf{Fire Containment}

We study the following model for fire containment. The input is a graph \(G = (V, E)\) with a source vertex \(s\), where the fire starts. At each time step we can choose up to \(k\) vertices to be saved by the firefighters, while the fire spreads to every vertex that has not been saved so far and has at least one burning neighbor. Once a vertex is saved or burns, it remains in this state permanently. The process stops when the fire cannot spread to any new vertices. This is a simple mathematical model for containing a natural process with a simple spread mechanism. It was first suggested by Hartnell [57] in the context of firefighting, and can also be used in a variety of similar scenarios, such as, for example, in containing an outbreak of a perfectly contagious disease via vaccination, when only a small number of individuals can
be vaccinated at each time step [55].

We consider the resource minimization version of the problem, called Resource Min-
mization Fire Containment (RMFC). In this problem, we are also given a subset \( T \subseteq V \) of
vertices called terminals. The goal is to minimize \( k \), the maximum number of vertices to be
saved at any time step, so that the fire does not spread to the vertices of \( T \). We sometimes
refer to \( k \) as the number of firefighters the solution uses.

King and MacGillivray [68] showed that RMFC is NP-hard even on full trees of degree
three, and their result implies that the problem is hard to approximate up to any factor
better than 2. This is the only currently known lower bound on the approximability of
RMFC, even for general graphs. On the algorithmic side, the standard randomized rounding
of a natural LP-relaxation for RMFC on trees gives an \( O(\log n) \)-approximation. King and
MacGillivray [68] also show that the problem is efficiently solvable on graphs of maximum
degree 3 if the fire starts at a degree-2 vertex.

Another natural version of the problem, that has been studied in the literature, is the
Firefighters problem, where the bound \( k \) is fixed, and the goal is to maximize the total
number of vertices to which the fire does not spread. The Firefighters problem on general
graphs are very hard to approximate. Indeed, Anshelevich et. al. [9] shows that Firefighters
problem in general graphs is \( n^{1-\epsilon} \)-hard to approximate for any \( \epsilon > 0 \), unless \( P = NP \).

Better approximation algorithms are known for the Firefighters problem on restricted
graph classes. On the trees, Hartnell and Li [56] showed that a simple greedy algorithm
gives a factor 2-approximation. This was improved to factor \( e/(e - 1) \) by to Cai, Verbin
and Yang [24], who also showed an exact algorithm with running time \( 2^{O(\sqrt{n} \log n)} \). Finbow
et. al. [49] showed that the Firefighters problem is NP-hard even on trees of degree at most
three, but is efficiently solvable if the fire starts at a degree-2 vertex. More recently, Fomin
et. al. [50] shows that for other various well-known graph classes, including interval graphs,
split graphs, permutation graphs, and \( P_k \)-free graphs for fixed \( k \), the Firefighters problem is
solvable in polynomial time.
There have also been some studies on the parameterized complexity of the problem. From a parameterized point of view, the Firefighters problem is W[1]-hard when parameterized by the number of saved vertices [17, 43].

**Related Work:** The model for fire spread control was introduced by Hartnell [57], who studied the fire containment on infinite graphs. Under his definition, given an infinite graph $G$, we say that the fire is contained if and only if only a finite number of vertices burn. We discuss some related works along this line here. Please refer to these papers and references therein for more detail. Wang and Moeller [83] showed that two firefighters are sufficient to contain the fire on $\mathbb{Z} \times \mathbb{Z}$ grid, and that any algorithm using two firefighters need at least 8 steps to do so. Develin and Hartke [44] later show that at least 18 vertices must burn in this scenario. Wang and Moeller [83] showed that $(r - 1)$ firefighters are sufficient to contain the fire in any $r$-regular graph, and in particular this implies that $(2d - 1)$ firefighters can contain the fire on a $d$-dimensional grid. This bound is tight, as shown in [44].

Another line of work that has received a lot of attention recently considers the “surviving rate of graphs” defined as follows. Given an input graph $G = (V, E)$, let the vertex $v$ denote the vertex where the fire starts. Denote by $\text{OPT}_k(v)$ the maximum number of vertices that can be protected when at most $k$ vertices can be saved at each time step. The surviving rate of graph $G$ is defined as an expected value of $\text{OPT}_k(v)$, when the vertex $v$ is chosen uniformly at random. Previous works mostly concern the surviving rate of special classes of graphs, such as planar graphs, trees, and bounded treewidth graphs. Please refer to, e.g. [69, 23, 84, 20] and references therein for more detail.

**Our Results**

We study the RMFC problem in various graph classes. Our main result is an $O(\log^* n)$ LP-rounding approximation algorithm for RMFC on trees. We also give an evidence that this result might be tight by showing a matching integrality gap of $\Omega(\log^* n)$ on the tree,
even when the LP is strengthened.

We show that our algorithmic technique for RMFC on trees also works when the input is a directed layered graph. We give an $O(\log n)$ approximation algorithm, which matches the integrality gap of natural LP relaxation.

We show that the problem seems to be very hard in general graphs by giving an $\Omega(n^{1/6})$ lower bound on the integrality gap of LP. However, using an $O(\log n)$ approximation algorithm for the tree case as a sub-routine, we can get $\tilde{O}(n^{1/3})$ approximation algorithm for RMFC on general graphs.

All our algorithmic results imply upper bounds on the LP integrality gap.
Part I

Integral Concurrent Flow
CHAPTER 1
PRELIMINARIES

In all our algorithmic results, we first solve the problem for the special case where all edge capacities are unit, and then extend our algorithms to general edge capacities. Therefore, in this section, we only discuss graphs with unit edge capacities.

1.1 Demands and Routing

Given any subset $S \subseteq V$ of vertices in graph $G$, denote by $\text{out}_G(S) = E_G(S, V \setminus S)$. We omit the subscript $G$ when clear from context. Let $\mathcal{P}$ be any collection of paths in graph $G$. We say that paths in $\mathcal{P}$ cause congestion $\eta$, iff for each edge $e \in E(G)$, the number of paths in $\mathcal{P}$ containing $e$ is at most $\eta$.

Given any graph $G = (V, E)$, and a set $T \subseteq V$ of terminals, a set $D$ of demands is a function $D : T \times T \rightarrow \mathbb{R}^+$, that specifies, for each unordered pair $t, t' \in T$ of terminals, a demand $D(t, t')$. We say that a set $D$ of demands is $\gamma$-restricted, iff for each terminal $t \in T$, the total demand $\sum_{t' \in T} D(t, t') \leq \gamma$. Given any partition $G$ of the terminals in $T$, we say that a set $D$ of demands is $(\gamma, G)$-restricted iff for each group $U \in G$, $\sum_{t \in U} \sum_{t' \in T} D(t, t') \leq \gamma$. We say that a demand set $D$ is integral iff $D(t, t')$ is integral for all $t, t' \in T$.

Given any set $D$ of demands, a fractional routing of $D$ is a flow $F$, where each pair $t, t' \in T$, of terminals sends $D(t, t')$ flow units to each other. Given an integral set $D$ of demands, an integral routing of $D$ is a collection $\mathcal{P}$ of paths, that contains $D(t, t')$ paths connecting each pair $(t, t')$ of terminals. The congestion of this integral routing is the congestion caused by the set $\mathcal{P}$ of paths in $G$. Any matching $M$ on the set $T$ of terminals defines an integral 1-restricted set $D$ of demands, where $D(t, t') = 1$ if $(t, t') \in M$, and $D(t, t') = 0$ otherwise. We do not distinguish between the matching $M$ and the corresponding set $D$ of demands.

Given any two subsets $V_1, V_2$ of vertices, we denote by $F : V_1 \rightsquigarrow_{\eta} V_2$ a flow from the vertices of $V_1$ to the vertices of $V_2$ where each vertex in $V_1$ sends one flow unit, and the
congestion due to $F$ is at most $\eta$. Similarly, we denote by $\mathcal{P} : V_1 \sim_{\eta} V_2$ a collection of paths $\mathcal{P} = \{ P_v \mid v \in V_1 \}$, where each path $P_v$ originates at $v$ and terminates at some vertex of $V_2$, and the paths in $\mathcal{P}$ cause congestion at most $\eta$. We define flows and path sets between subsets of edges similarly. For example, given two collections $E_1, E_2$ of edges of $G$, we denote by $F : E_1 \sim_{\eta} E_2$ a flow that causes congestion at most $\eta$ in $G$, where each flow-path has an edge in $E_1$ as its first edge, and an edge in $E_2$ as its last edge, and moreover each edge in $E_1$ sends one flow unit (notice that it is then guaranteed that each edge in $E_2$ receives at most $\eta$ flow units due to the bound on congestion). We will often be interested in a scenario where we are given a subset $S \subseteq V(G)$ of vertices, and $E_1, E_2 \subseteq \text{out}(S)$. In this case, we say that a flow $F : E_1 \sim_{\eta} E_2$ is contained in $S$, iff for each flow-path $P$ in $F$, all edges of $P$ belong to $G[S]$, except for the first and the last edges that belong to $\text{out}(S)$. Similarly, we say that a set $\mathcal{P} : E_1 \sim_{\eta} E_2$ of paths is contained in $S$, iff all inner edges on paths in $\mathcal{P}$ belong to $G[S]$.

**Edge-Disjoint Paths:** We use the algorithm of [40] for EDP as a subroutine. The following theorem of [40] is summarized below.

**Theorem 5 ([40])** Let $G$ be any graph with unit edge capacities and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs. Assume further that there is a multicommodity flow where the pairs in $\mathcal{M}$ altogether send $\text{OPT}$ flow units to each other, with no congestion, and at most one flow unit is sent between each pair. Then there is an efficient randomized algorithm that w.h.p. finds a collection $\mathcal{P}$ of paths, connecting at least $\text{OPT}/\alpha_{\text{EDP}}$ of the demand pairs, such that the congestion of $\mathcal{P}$ is at most $\eta_{\text{EDP}} = 14$, where $\alpha_{\text{EDP}} = \text{poly log } k$.

## 1.2 Sparsest Cut and the Flow-Cut Gap

Given a graph $G = (V, E)$ with a subset $T$ of vertices called terminals, the sparsity of any partition $(A, B)$ of $V$ is $\frac{|E(A,B)|}{\min\{|A\cap T|, |B\cap T|\}}$. The goal of the sparsest cut problem is to find a partition $(A, B)$ of $V$ with minimum sparsity. Arora, Rao and Vazirani [12] have shown an
$O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We denote by $A_{ARV}$ this algorithm and by $\alpha_{ARV}(k) = O(\sqrt{\log k})$ its approximation factor.

Sparsest cut is the dual of the Maximum Concurrent Flow problem, where for each pair $(t, t')$ of terminals, the demand $D(t, t') = 1/k$. The maximum possible ratio, in any graph, between the value of the minimum sparsest cut and the value $\lambda$ of the maximum concurrent flow, is called the flow-cut gap. The flow-cut gap in undirected graphs, that we denote by $\beta_{FCG}(k)$ throughout the paper, is $\Theta(\log k)$ [71, 53, 73, 14]. In particular, if the value of the sparsest cut in graph $G$ is $\alpha$, then every pair of terminals can send at least $\frac{\alpha}{k \beta_{FCG}(k)}$ flow units to each other simultaneously with no congestion. Moreover, any 1-restricted set $D$ of demands on the set $T$ of terminals can be fractionally routed with congestion at most $2 \beta_{FCG}(k)/\alpha$ in $G$.

1.3 Well-linkedness

Given any subset $S \subseteq V$ of vertices, we say that $S$ is $\alpha$-well-linked, iff for any partition $(A, B)$ of $S$, if we denote $T_A = \text{out}(S) \cap \text{out}(A)$ and $T_B = \text{out}(S) \cap \text{out}(B)$, then $|E(A, B)| \geq \alpha \cdot \min\{|T_A|, |T_B|\}$.

Given a subset $S$ of vertices, we can sub-divide every edge $e \in \text{out}(S)$ by a vertex $z_e$, and set $T' = \{z_e \mid e \in \text{out}(S)\}$. Let $G_S$ be the sub-graph of the resulting graph induced by $S \cup T'$. Then $S$ is $\alpha$-well-linked (for $\alpha \leq 1$) in $G$ iff the value of the sparsest cut in graph $G_S$ for the set $T'$ of terminals is at least $\alpha$. In particular, if $S$ is $\alpha$-well-linked, then any 1-restricted set $D$ of demands on $\text{out}(S)$ can be fractionally routed inside $S$ with congestion at most $2 \beta_{FCG}(k)/\alpha$.

Similarly, given any graph $G = (V, E)$ with a subset $T$ of vertices called terminals, we say that $G$ is $\alpha$-well-linked for $T$ iff for any partition $(A, B)$ of $V$, $|E_G(A, B)| \geq \alpha \cdot \min\{|T \cap A|, |T \cap B|\}$.

Let $L = \text{poly log } n$ be a parameter to be fixed later. (We use different values of $L$ in different algorithms.) We say that a cluster $X \subseteq V(G)$ is small iff $|\text{out}(X)| \leq L$, and we
say that it is large otherwise.

A useful tool in graph routing algorithms is a well-linked decomposition [36, 77]. This is a procedure that, given any subset $S$ of vertices, produces a partition $\mathcal{W}$ of $S$ into well-linked subsets. In the following theorem we describe a new well-linked decomposition. In addition to the standard properties guaranteed by well-linked decompositions, we obtain a collection of paths connecting the edges of $\bigcup_{R \in \mathcal{W}} \text{out}(R)$ to the edges of $\text{out}(S)$, with a small congestion.

**Theorem 6 (Extended well-linked decomposition)** There is an efficient algorithm, that, given any set $S$ of vertices, with $|\text{out}(S)| = k'$, produces a partition $\mathcal{W}$ of $S$ with the following properties.

- For each set $R \in \mathcal{W}$, $|\text{out}(R)| \leq k'$. If $R$ is a large cluster, then it is $\alpha_{\mathcal{W}} = \Omega(1/\log^{1.5}n)$-well-linked. If $R$ is a small cluster, then it is $\alpha_S = \Omega(1/(\log \log n)^{1.5})$-well-linked.

- Let $E^* = (\bigcup_{R \in \mathcal{W}} \text{out}(R)) \setminus \text{out}(S)$. Then we can efficiently find a set $N = \{\tau_e \mid e \in E^*\}$ of paths called tendrils contained in $G[S]$, where tendon $\tau_e$ connects edge $e$ to some edge of $\text{out}(S)$, each edge in $\text{out}(S)$ participates in at most one tendon, and the total congestion caused by $N$ is at most 3.

- $|E^*| \leq 0.4|\text{out}(S)|$.

The main difference of this well-linked decomposition from the standard one is that it is “one-shot”: given a current set $S$ of vertices, we iteratively find a subset $R \subseteq S$, where $R$ is well-linked itself, add $R$ to the partition $\mathcal{W}$, update $S$ to be $S \setminus R$, and then continue with the updated set $S$. Unlike the standard well-linked decompositions, we do not recurse inside the set $R$, which is guaranteed to be well-linked. We will still be able to bound the number of partition edges $\sum_{R \in \mathcal{W}} |\text{out}(R)|$ in terms of $|\text{out}(S)|$ as before, and moreover construct the tendrils connecting every edge in $\bigcup_{R \in \mathcal{W}} \text{out}(R)$ to the edges of $\text{out}(S)$ with constant congestion.
We start with some definitions. We assume that we are given a graph $G$, and a subset $S \subseteq V(G)$, with $|\text{out}(S)| = k'$. Given any integer $r$, let $\alpha(r) = \frac{1}{10} \left(1 - \frac{1}{\log k'}\right)^{\log r}$. Notice that for $1 \leq r \leq k'$, $\frac{1}{20e} \leq \alpha(r) \leq \frac{1}{10}$.

**Definition 1** Let $R$ be any subset of vertices of $S$, and let $(X, Y)$ be any partition of $R$, with $T_X = \text{out}(R) \cap \text{out}(X)$, $T_Y = \text{out}(R) \cap \text{out}(Y)$, and $|T_X| \leq |T_Y|$. We say that $X$ is a sparse cut for $R$, if

$$|E_G(X, Y)| < \alpha(r)|T_X|,$$

where $r = |T_X|$.

Observe that if $X$ is a sparse cut for $S$, and $Y = S \setminus X$, then $|\text{out}(X)|, |\text{out}(Y)| \leq |\text{out}(S)|$. Notice also that since $\alpha(r) \geq \frac{1}{20e}$, if set $S$ has no sparse cuts, then it is $\frac{1}{20e}$-well-linked.

We now proceed as follows. First, we obtain a non-constructive well-linked decomposition, that gives slightly weaker guarantees. In particular, it does not ensure that the small clusters are $\alpha_S$-well-linked. We then turn this decomposition into a constructive one, while slightly weakening its parameters. Finally, we show an algorithm, that improves the parameters of the decomposition, by recursively applying it to some of the clusters.

### 1.3.1 Non-constructive decomposition

We show an algorithm to construct the decomposition, whose running time is exponential in $k'$. We later turn this algorithm into an efficient one, with slightly worse parameters. We note that in the current section we do not ensure that every small cluster is $\alpha_S$-well linked, but we obtain this guarantee in our final decomposition. The decomposition algorithm works as follows.

- Start with $\mathcal{W} = \emptyset$, $S' = S$. 

• While $S'$ contains any sparse cut:
  
  – Let $X$ be a sparse cut minimizing $|T_X|$; if there are several such cuts, choose one
    minimizing the number of vertices $|X|$.
  
  – Add $X$ to $W$ and set $S' := S' \setminus X$.

• Add $S'$ to $W$

We now proceed to analyze the algorithm. First, we show that all sets added to $W$ are
well-linked, in the next lemma.

**Lemma 1** Every set added to $W$ is $\frac{1}{80e \log k'}$-well linked.

**Proof:** First, the last set added to $W$ must be $\frac{1}{20e}$-well-linked, since $S'$ does not contain
any sparse cuts at this point.

Consider now some iteration of the algorithm. Let $S'$ be the current set, $X$ the sparse cut
we have selected, and $Y = S' \setminus X$. We denote $T_X = \text{out}(X) \cap \text{out}(S')$, $T_Y = \text{out}(Y) \cap \text{out}(S')$, $\Gamma = E_{G}(X,Y)$, $r = |T_X|$, and recall that $|T_X| \leq |T_Y|$, and $|\Gamma| < r \cdot \alpha(r)$.

![Figure 1.1: Illustration for Lemma 1](image)

We now show that $X$ is $\frac{1}{80e \log k'}$-well linked. Let $(A, B)$ be any partition of $X$. We can
assume that $A$ and $B$ are not sparse cuts for $S'$: otherwise, we should have added $A$ or $B$ to $W$ instead of $X$. The following claim will then finish the proof.
Claim 1 If \( X \) is a sparse cut for \( S' \), \((A,B)\) is a partition of \( X \), and \( A,B \) are not sparse cuts for \( S' \), then

\[
|E(A,B)| \geq \frac{1}{80e \cdot \log k'} \min \{ |\text{out}(A) \cap \text{out}(X)|, |\text{out}(B) \cap \text{out}(X)| \}.
\]

Proof: We denote \( T'_X = T_X \cap \text{out}(A) \), \( T''_X = T_X \cap \text{out}(B) \), \( \Gamma' = \Gamma \cap \text{out}(A) \), \( \Gamma'' = \Gamma \cap \text{out}(B) \), and we assume w.l.o.g. that \( |T'_X| \leq |T''_X| \) (notice that it is possible that \( T'_X = \emptyset \)). Let \( E' = E(A,B) \), \( |T'_X| = r' \), and \( |T''_X| = r'' \) (see Figure 1.2).

Since \( B \) is not sparse for \( S' \), \( |E'| + |\Gamma''| \geq r'' \cdot \alpha(r'') \), and equivalently, \( |\Gamma''| \geq r'' \cdot \alpha(r'') - |E'| \) (since \( r'' \leq r \) and so \( \alpha(r'') \geq \alpha(r) \)).

On the other hand, since \( X \) is a sparse cut for \( S' \), \( |\Gamma'| + |\Gamma''| < r \cdot \alpha(r) \), so \( |\Gamma'| < r' \alpha(r) + |E'| \). Notice that if \( T'_X = \emptyset \), then \( |\Gamma'| < |E'| \) must hold, contradicting the assumption that \((A,B)\) is a violating cut. Therefore, we assume from now on that \( T'_X \neq \emptyset \).

Recall that \( r' \leq r/2 \), and so \( \log r' \leq \log r - 1 \). Therefore,

\[
\alpha(r') = \frac{1}{10} \left( 1 - \frac{1}{\log k'} \right)^{\log r'} \geq \frac{1}{10} \left( 1 - \frac{1}{\log k'} \right)^{\log r - 1} = \alpha(r) / \left( 1 - \frac{1}{\log k'} \right)
\]

So \( \alpha(r) \leq \left( 1 - \frac{1}{\log k'} \right) \cdot \alpha(r') \). We therefore get that:
\[ |\Gamma'| < r'\alpha(r') \cdot \left(1 - \frac{1}{\log k'}\right) + |E'| \]  

(1.1)

Since \( A \) is not a sparse cut for \( S' \), \( |E'| + |\Gamma'| \geq r' \cdot \alpha(r') \), and \( |\Gamma'| \geq r' \cdot \alpha(r') - |E'| \).

Combining this with Equation (1.1), we get that

\[ 2|E'| + r'\alpha(r') \cdot \left(1 - \frac{1}{\log k'}\right) > r' \cdot \alpha(r'), \]

and rearranging the sides:

\[ |E'| > \frac{r'\alpha(r')}{{2\log k'}} \]  

(1.2)

Substituting this in Equation (1.1), we get that:

\[ |\Gamma'| < r'\alpha(r') \cdot \left(1 - \frac{1}{\log k'}\right) + |E'| < r'\alpha(r') + |E'| < 3|E'| \log k' \]

and so

\[ |E'| > \frac{|\Gamma'|}{3\log k'} \]  

(1.3)

Combining Equations (1.2) and (1.3), we get that

\[ |E'| > \frac{|\Gamma'|}{6\log k'} + \frac{r'\alpha(r')}{4\log k'} > \frac{|\Gamma'| + r'}{80e \cdot \log k'}. \]

We conclude that

\[ |E(A, B)| \geq \frac{1}{80e \cdot \log k'} |\text{out}(A) \cap \text{out}(X)|. \]

\[ \square \]

**Constructing the tendrils**  The next lemma will be useful in constructing the tendrils.
Lemma 2 Let $X$ be any sparse cut added to $\mathcal{W}$ during the execution of the algorithm, $T_X = \text{out}(X) \cap \text{out}(S')$, where $S'$ is the current set, and $\Gamma = E(X, S' \setminus X)$. Then there is a flow $F$ in $G[X]$ with the following properties:

- Each edge in $\Gamma$ sends one flow unit.
- Each edge in $T_X$ receives $1/10$ flow unit.
- Congestion at most 1.

Proof: We set up the following single source-sink network. Start with graph $G$, and for each edge $e \in \text{out}(X)$, subdivide $e$ by a vertex $t_e$. Let $H$ be the sub-graph of the resulting graph induced by $X \cup \{t_e | e \in \text{out}_G(X)\}$. Unify all vertices $t_e$ for $e \in T_X$ into a source $s$, and unify all vertices $t_e$ with $e \in \Gamma$ into a sink $t$. The capacities of all edges are unit, except for the edges adjacent to the source, whose capacities are $1/10$. Let $H'$ denote the resulting flow network. The existence of the required flow in $G[X]$ is equivalent to the existence of an $s$-$t$ flow of value $|\Gamma|$ in $H'$.

Assume for contradiction that such flow does not exist. Then there is a cut $(A, B)$ in $H'$ with $s \in A$, $t \in B$, and $|E(A, B)| < |\Gamma|$. Let $T_1 = E(A, B) \cap T_X$, $T_2 = T_X \setminus T_1$, and similarly, $\Gamma_1 = E(A, B) \cap \Gamma$, $\Gamma_2 = \Gamma \setminus \Gamma_1$. Let $E' = E(A, B) \setminus (\Gamma_1 \cup T_1)$.

![Figure 1.3: Illustration for Lemma 2](image)

We then have that $|T_1|/10 + |\Gamma_1| + |E'| < |\Gamma|$. We claim that $(A \setminus \{s\})$ is a sparse cut for $S'$, and so it should have been chosen instead of $X$. Indeed, since $|T_1|/10 < |\Gamma|$, and
$|T_X| \geq 10|\Gamma|$, $T_2 \neq \emptyset$.

Let $|T_1| = r'$, $|T_2| = r''$. In order to prove that $(A \setminus \{s\})$ is a sparse cut, it is enough to prove that:

$$|E'| + |\Gamma_1| < r''\alpha(r'').$$

Recall that $|E'| + |\Gamma_1| < |\Gamma| - r'/10 < r\alpha(r) - r'/10$, since $|\Gamma| < r \cdot \alpha(r)$, because $X$ is sparse for $S'$. Since $\alpha(r) \leq \frac{1}{10}$, we get that $r'/10 \geq r'\alpha(r)$. We conclude that $|E'| + |\Gamma_1| < r\alpha(r) - r'\alpha(r) = r''\alpha(r) \leq r''\alpha(r'')$, a contradiction. \(\square\)

**Corollary 1** Let $W$ be the final partition of $S$ produced by the algorithm, and let $E^* = (\bigcup_{R \in W} \text{out}(R)) \setminus \text{out}(S)$ be the set of the new edges of the partition. Then there is a flow $F : E^* \rightsquigarrow \text{out}(S)$ in graph $G[S] \cup \text{out}(S)$, where each edge in $E^*$ sends one flow unit, each edge in $\text{out}(S)$ receives at most 0.2 flow units, and the total congestion is at most 1.2.

**Proof:** For each edge $e \in E^* \cup \text{out}(S)$, we will define a flow $F_e$, connecting $e$ to the edges in $\text{out}(S)$, and the final flow $F$ will be the union of these flows over all edges $e \in E^*$. At the beginning, each edge $e \in \text{out}(S)$ sends one flow unit to itself.

We now follow the decomposition algorithm, and maintain the following invariant: at each step of the algorithm, for each edge $e \in (\bigcup_{R \in W} \text{out}(R))$, the flow $F_e$ is already defined. In particular, if $S'$ is the current set, then for each edge $e \in \text{out}(S')$, the flow $F_e$ is already defined. Consider a step of the algorithm, where we find a sparse cut $X$ for $S'$, and add $X$ to $W$. Let $\Gamma = E(X, S' \setminus X)$, and let $T_X = \text{out}(X) \cap \text{out}(S')$. From Lemma 2, there is a flow $F_1 : \Gamma \rightsquigarrow T_X$, where each edge $e \in \Gamma$ sends one flow unit, each edge in $T_X$ receives at most 0.1 flow unit, the congestion is 1, and the flow is contained in $G[X]$. Consider some edge $e \in \Gamma$. Flow $F_e$ is defined as follows: we use all flow-paths originating in $e$ in $F_1$. For each such flow-path $P$, if $e' \in T_X$ is the other endpoint of $P$, then we concatenate $P$ with $F_{e'}$, scaled down by factor $c(e') \leq 0.1$, where $c(e')$ is the total flow that edge $e'$ receives in $F_1$. Notice that $F_e$ is a valid unit flow from $e$ to $\text{out}(S)$. 23
This completes the description of the flow \( F \). In order to bound the amount of flow that each edge \( e \in \text{out}(S) \) receives, notice that this flow forms a geometric progression: namely, if \( E_1 \) is the set of edges that send flow directly to \( e \), \( E_2 \) is the set of edges that send flow directly to edges of \( E_1 \), and so on, then the total flow that \( e \) receives from \( E_1 \) is at most 0.1, the total flow that it receives from \( E_2 \) is at most 0.01 and so on. Therefore, the total flow that every edge in \( \text{out}(S) \) receives is bounded by 0.2. Congestion on edges is bounded by 1.2 similarly.

The next corollary follows from Corollary 1 and the integrality of flow.

**Corollary 2** There is a collection \( N = \{\tau_e | e \in E^*\} \) of paths (called tendrils) in graph \( G[S] \), where each path \( \tau_e \) connects \( e \) to some edge in \( \text{out}(S) \), the total congestion caused by path in \( N \) is at most 2, and each edge in \( S \) serves as an endpoint of at most one tendril.

### 1.3.2 Constructive Version

In order to obtain an efficient algorithm for finding the decomposition, we use an approximation algorithm \( \mathcal{A}_{\text{ARV}} \) for the Sparsest Cut problem. As before, we start with \( \mathcal{W} = \emptyset \), and \( S' = S \). We then perform a number of iterations, which are executed as follows. We set up an instance of the sparsest cut problem on graph \( G[S'] \), with the set \( \text{out}(S') \) of edges acting as terminals (imagine placing a terminal \( t_e \) on each edge \( e \in \text{out}(S') \)). We then apply algorithm \( \mathcal{A}_{\text{ARV}} \) to the resulting instance of the sparsest cut problem. Let \((X,Y)\) be the output of algorithm \( \mathcal{A}_{\text{ARV}} \), and assume w.l.o.g. that \( |\text{out}(X) \cap \text{out}(S')| \leq |\text{out}(Y) \cap \text{out}(S')| \).

If \( |E(X,Y)| \geq \frac{1}{20e} |\text{out}(A) \cap \text{out}(S')| \), then we are guaranteed that set \( S' \) is \( \frac{1}{20e \cdot \alpha_{\text{ARV}}(k')} \)-well linked. In this case, we simply add \( S' \) to \( \mathcal{W} \), and finish the algorithm.

Assume now that \( |E(X,Y)| < \frac{1}{20e} |\text{out}(X) \cap \text{out}(S')| \). Clearly, this means that \( X \) is a sparse cut for \( S' \). Let \( T_X = \text{out}(X) \cap \text{out}(S') \), and \( \Gamma = E(X,Y) \). We say that a flow \( F : \Gamma \leadsto T_X \) is a good flow iff every edge in \( \Gamma \) sends one flow unit, every edge in \( T_X \) receives at most 0.1 flow units, the flow is contained in \( G[X] \), and it causes congestion at most 1. If we are to add \( X \) to \( \mathcal{W} \), we would like to ensure that \( X \) is well-linked, and a good flow
exists for $X$. In the next theorem we show that if this is not the case, then we can find another sparse cut $A \subseteq X$ for $S'$, such that either $|\text{out}(A) \cap \text{out}(S')| < |\text{out}(X) \cap \text{out}(S')|$, or $|\text{out}(A) \cap \text{out}(S')| = |\text{out}(X) \cap \text{out}(S')|$ and $|A| < |X|$. 

Lemma 3 Assume that $X$ is a sparse cut for $S'$. Then either $X$ is $\frac{1}{80e \log k' \alpha_{ARV}(k')}$-well-linked and has a good flow $F$, or we can efficiently find another sparse cut $A \subseteq X$ for $S'$, such that either $|\text{out}(A) \cap \text{out}(S')| < |\text{out}(X) \cap \text{out}(S')|$, or $|\text{out}(A) \cap \text{out}(S')| = |\text{out}(X) \cap \text{out}(S')|$ and $|A| < |X|$. 

We prove the lemma below, but we first complete the description of the algorithm. Let $\alpha_W(k') = \frac{1}{80e \log k' \alpha_{ARV}(k')}$. Once we find a sparse cut $X$ for $S'$, we apply Lemma 3 to $X$. If the outcome is that $X$ is well-linked and has a good flow, then we simply add $X$ to $W$, set $S' := S' \setminus X$, and continue to the next iteration of the decomposition. Otherwise, if we have found a sparse cut $A \subseteq X$ as above, then we replace $X$ by $A$, and repeat this step again. We continue applying Lemma 3 to the current set $X$, until we obtain a set that is $\alpha_W(k')$-well-linked and contains a good flow. We then add $X$ to $W$. Since at every step we are guaranteed that either $|T_X|$ decreases, or $|T_X|$ stays the same but $|X|$ decreases, we are guaranteed that after at most $k'n$ applications of Lemma 3 we will obtain a set $X$ that can be added to $W$. We now give a proof of Lemma 3.

Proof: [of Lemma 3] The proof consists of two steps. First, we try to find a good flow from $\Gamma$ to $T_X$ exactly as in Lemma 2. We build a flow network $H'$ exactly as in the proof of Lemma 2. If such flow does not exist, then we obtain a cut $(A, B)$ as before. From the proof of Lemma 3, in this case $A$ is a sparse cut for $S'$, and moreover either $|\text{out}(A) \cap \text{out}(S')| < |\text{out}(X) \cap \text{out}(S')|$, or $|\text{out}(A) \cap \text{out}(S')| = |\text{out}(X) \cap \text{out}(S')|$ and $|A| < |X|$. So if $X$ does not contain a good flow, we can return a cut $A$ as required. If $X$ contains a good flow, then we perform the following step.

We run the algorithm $A_{ARV}$ on the graph $G[X]$, where the edges of out$(X)$ serve as
terminals. Let \((A, B)\) be the resulting cut. If

\[ |E(A, B)| \geq \frac{1}{80e \cdot \log k'} \min \{|\text{out}(A) \cap \text{out}(X)|, |\text{out}(B) \cap \text{out}(X)|\} \]

, then we are guaranteed that \(X\) is \(\frac{1}{80e \cdot \log k'}\)-well-linked as required. Otherwise, from the proof of Claim 1, either \(A\) or \(B\) must be a sparse cut for \(S'\), satisfying the conditions of the lemma.

We summarize the algorithm, which we call from now on the basic well-linked decomposition, in the next theorem.

**Theorem 7 (Basic well-linked decomposition)** There is an efficient algorithm, that, given a set \(S\) of vertices, with \(|\text{out}(S)| = k'\), produces a partition \(W\) of \(S\) with the following properties.

- For each set \(R \in W\), \(|\text{out}(R)| \leq k'\) and \(R\) is \(\alpha_W(k') = \Omega\left(\frac{1}{\log^{1.5} k'}\right)\)-well-linked.
- Let \(E^* = (\bigcup_{R \in W} \text{out}(R)) \setminus \text{out}(S)\). Then there is a flow \(F : E^* \rightsquigarrow \text{out}(S)\), in \(G[S]\), where each edge in \(E^*\) sends one flow unit, each edge in \(\text{out}(S)\) receives at most 0.2 flow units, and the congestion is at most 1.2.

As before, from the integrality of flow, we can build a set \(N = \{\tau_e \mid e \in E^*\}\) of tendrils contained in \(G[S]\), where each tendril \(\tau_e\) connects edge \(e\) to some edge of \(\text{out}(S)\), each edge in \(\text{out}(S)\) participates in at most one tendril, and the total congestion caused by \(N\) is at most 2.

**Extended Well-Linked Decomposition**

We would like to obtain a well-linked decomposition similar to Theorem 7 with one additional property: if \(R \in W\) is a small cluster, then it is \(\Omega(1/(\log \log n)^{1.5})\)-well-linked.

In order to achieve this, given a set \(S\), we first perform a basic well-linked decomposition as in Theorem 7. Let \(W\) be the resulting decomposition, \(E^* = (\bigcup_{R \in W} \text{out}(R)) \setminus \text{out}(S),\)
and let $F : E^* \leadsto \text{out}(S)$ be the resulting flow.

For each small cluster $R \in \mathcal{W}$, we then perform another round of basic well-linked decomposition on the set $R$ (notice that now $k' = |\text{out}(R)| \leq L$). Let $\mathcal{W}_R$ be the resulting partition, $E^*_R$ the resulting set of partition edges (excluding the edges of out($R$)), and $F^R$ the resulting flow.

We build the final partition $\mathcal{W}'$ as follows. Start from $\mathcal{W}$, and replace each small set $R \in \mathcal{W}$ by the sets in $\mathcal{W}_R$. Let $E^{**} = \left( \bigcup_{R \in \mathcal{W}} \text{out}(R) \right) \setminus \text{out}(S)$. We extend the flow $F : E^* \leadsto \text{out}(S)$ to flow $F' : E^{**} \leadsto \text{out}(S)$ as follows. Consider a small set $R \in \mathcal{W}$. Recall that we have defined a flow $F^R : E^*_R \leadsto \text{out}(R)$ inside $R$, where each edge in $E^*_R$ sends one flow unit, and each edge in out($R$) receives at most 0.2 flow units. We concatenate this flow with the flow originating from the edges of out($R$) in $F$ (after scaling it appropriately, by the factor of at most 0.2). After we process all small sets $R \in \mathcal{W}$, we obtain the final flow $F'$. Every edge in $E^{**}$ now sends one flow unit, every edge in out($S$) receives at most 0.4 flow units, and total congestion is at most 3. As before, from the integrality of flow, we can build a set $N = \{ \tau_e | e \in E^{**} \}$ of tendrils, where each tendril $\tau_e$ connects edge $e$ to some edge of out($S$), each edge in out($S$) participates in at most one tendril, and the total congestion caused by $N$ is at most 3. This concludes the proof of Theorem 6.

1.3.3 The Grouping Technique

The grouping technique was first introduced by Chekuri, Khanna and Shepherd [35], and it is widely used in algorithms for network routing [36, 79, 5], in order to boost network connectivity and well-linkedness parameters. We summarize it in the following theorem.

**Theorem 8** Suppose we are given a connected graph $G = (V, E)$, with weights $w(v)$ on vertices $v \in V$, and a parameter $p$. Assume further that for each $v \in V$, $0 \leq w(v) \leq p$, and $\sum_{v \in V} w(v) \geq p$. Then we can find a partition $\mathcal{G}$ of the vertices in $V$, and for each group $U \in \mathcal{G}$, find a tree $T_U \subseteq G$ containing all vertices of $U$, such that for each group $U \in \mathcal{G}$, $p \leq w(U) \leq 3p$, where $w(U) = \sum_{v \in U} w(v)$, and the trees $\{T_U\}_{U \in \mathcal{G}}$ are edge-disjoint.
1.4 Bandwidth Property and Critical Clusters

Given a graph $G$, and a subset $S$ of vertices of $G$ we say that the modified bandwidth condition holds for $S$, iff $S$ is $\alpha_{bw}$-well-linked if it is a large cluster, and it is $\alpha_S$-well-linked if it is a small cluster, where $\alpha_S = \Omega \left( \frac{1}{(\log \log n)^{1.5}} \right)$, and $\alpha_{bw} = \alpha_W \cdot \alpha_S = \Omega \left( \frac{1}{(\log n \log \log n)^{1.5}} \right)$. For simplicity, we will use “bandwidth property” instead of “modified bandwidth property” from now on.

Given a subset $S$ of vertices of $G$, and a partition $\pi$ of $S$, let $H_S$ be the following graph: start with $G[S]$, and contract each cluster $C \in \pi$ into a super-node $v_C$. Set the weight $w(v_C)$ of $v_C$ to be $|\text{out}_{G}(C)|$ (notice that the weight takes into account all edges incident on $C$, including those in $\text{out}(S)$). We use the parameter $\lambda = \frac{\alpha_{bw}}{8\alpha_{ARV}(n)} = \Omega \left( \frac{1}{\log^2 n \cdot (\log \log n)^{1.5}} \right)$.

**Definition 2** Given a subset $S$ of vertices of $G$ and a partition $\pi$ of $S$, we say that $(S, \pi)$ has the weight property with parameter $\lambda'$, iff for any partition $(A, B)$ of $V(H_S)$, $|E_{H_S}(A, B)| \geq \lambda' \cdot \min \left\{ \sum_{v \in A} w(v), \sum_{v \in B} w(v) \right\}$. If the weight property holds for the parameter $\lambda = \lambda'$, then we simply say that $(S, \pi)$ has the weight property.

**Definition 3** Given a subset $S$ of vertices and a partition $\pi$ of $S$, we say that $S$ is a critical cluster iff (1) $S$ is a large cluster and it has the bandwidth property; (2) Every cluster $R \in \pi$ is a small cluster and it has the bandwidth property; and (3) $(S, \pi)$ has the weight property. Additionally, if $S = \{v\}$, and the degree of $v$ is greater than $L$, then we also say that $S$ is a critical cluster.

Let $\eta^* = \frac{2\beta_{FCG}(L)}{\alpha_S} = O((\log \log n)^{2.5})$. We say that a cut $(S, \overline{S})$ in $G$ is large iff $|E(S, \overline{S})| \geq \frac{L}{4\eta^*}$. Note that we somewhat abuse the notation: We will say that a cluster $S$ is large iff $|\text{out}(S)| > L$, but we say that a cut $(S, \overline{S})$ is large iff $|E(S, \overline{S})| \geq \frac{L}{4\eta^*}$.

In the next lemma we show that if we are given any large cluster $S$ that has the bandwidth property, then we can find a critical sub-cluster $Q$ of $S$. Moreover, there is a subset of at least $L/4$ edges of $\text{out}(Q)$ that can be routed to the edges of $\text{out}(S)$. One can prove a
similar lemma using the Räcke decomposition as a black-box. Since we use slightly different parameters in the definitions of small and critical clusters, we prove the lemma directly.

**Lemma 4** Let $S$ be any large cluster that has the bandwidth property. Then we can efficiently find a critical cluster $Q \subseteq S$, a subset $E_Q \subseteq \text{out}(Q)$ of $L/4$ edges, and a set $P_Q : E_Q \sim \eta^*$ of paths, which are contained in $S \setminus Q$, such that for each edge $e \in \text{out}(S)$, at most one path of $P_Q$ terminates at $e$.

### 1.4.1 Proof of Lemma 4

Let $G'$ be a graph obtained from $G$ as follows: subdivide every edge $e \in \text{out}(S)$ with a vertex $v_e$, and let $T' = \{v_e \mid e \in \text{out}(S)\}$. Graph $G'$ is the sub-graph of $G$ induced by $T' \cup S$.

Throughout the algorithm, we maintain a collection $\pi$ of disjoint subsets of vertices of $S$, together with a corresponding contracted graph $Z$, which is obtained from graph $G'$, by contracting every cluster $C \in \pi$ into a super-node $v_C$. We say that $\pi$ is a *good collection of clusters*, iff each cluster $C \in \pi$ is small and has the bandwidth property. The value $W(\pi)$ of the collection $\pi$ of clusters is the number of edges in the corresponding contracted graph $Z$.

We notice that some vertices of $S$ may not belong to any cluster in $\pi$. Our initial collection is $\pi = \emptyset$.

We say that a cluster $S' \subseteq S$ is *canonical* for the collection $\pi$ iff for every cluster $C \in \pi$, either $C \subseteq S'$, or $C \cap S' = \emptyset$.

Throughout the algorithm, we also maintain an active large cluster $S' \subseteq S$ (the initial cluster $S' = S$). We will ensure that $S'$ is canonical w.r.t. the current collection $\pi$ of good clusters, and it has the bandwidth property. We perform a number of iterations. In each iteration, one of the following three things happens: we either find a new good collection $\pi'$ of clusters, with $W(\pi') < W(\pi)$, or find a critical cluster $Q$ as required, or select a sub-cluster $S'' \subsetneq S'$ as our next active cluster. In the latter case, we will guarantee that $S''$ is canonical for the current collection $\pi$ of good clusters, and it has the bandwidth property. An execution
of an iteration is summarized in the next lemma whose proof appears in the Appendix ??.

The proof uses arguments similar in spirit to the analysis of the Räcke decomposition [77].

**Lemma 5** Let \( \pi \) be a good collection of clusters, and let \( S' \subseteq S \) be a large cluster with the bandwidth property, such that \( S' \) is canonical for \( \pi \). Assume additionally that there is a set \( E_{S'} \subseteq \text{out}(S') \) of \( L/4 \) edges, and a set \( \mathcal{P}_{S'} : E_{S'} \sim_{\eta^*} \text{out}(S) \) of paths in graph \( G \), contained in \( S \setminus S' \), where each edge in \( \text{out}(S) \) is an endpoint of at most one path. Then there is an efficient algorithm, whose output is one of the following:

- Either a good collection \( \pi' \) of clusters with \( W(\pi') < W(\pi) \).
- Or establishes that \( S' \) is a critical cluster, by computing, if \( |S'| > 1 \), a partition \( \pi^* \) of \( S' \) into small clusters that have bandwidth property, such that \( (S', \pi^*) \) has the weight property.
- Or a sub-cluster \( S'' \subseteq S' \), such that \( S'' \) is large, canonical for \( \pi \), has the bandwidth property, and there is a set \( E_{S''} \subseteq \text{out}(S'') \) of \( L/4 \) edges, and a set \( \mathcal{P}_{S''} : E_{S''} \sim_{\eta^*} \text{out}(S) \) of paths in graph \( G \), contained in \( S \setminus S'' \), where each edge in \( \text{out}(S) \) is an endpoint of at most one path.

We now complete the proof of Lemma 4. We start with \( S' = S \) and an initial collection \( \pi = \emptyset \). We then iteratively apply Lemma 5 to the current cluster \( S' \) and the current partition \( \pi \). If the lemma returns a good collection \( \pi' \) of clusters, whose value \( W(\pi') \) is smaller than the value \( W(\pi) \) of \( \pi \), then we replace \( \pi \) with \( \pi' \), set the current active cluster to be \( S' = S \), and continue. Otherwise, if it returns a sub-cluster \( S'' \subseteq S' \), then we replace \( S' \) with \( S'' \) as the current active cluster and continue. Finally, if it establishes that \( S' \) is a critical cluster, then we return \( S', \pi^* \), the set \( E_{S'} \) of edges, and the collection \( \mathcal{P}_{S'} \) of paths. It is easy to verify that the algorithm terminates in polynomial time: we partition the algorithm execution into phases. A phase starts with some collection \( \pi \) of clusters, and ends when we obtain a new collection \( \pi' \) with \( W(\pi') < W(\pi) \). Clearly, the number of phases is bounded by \( |E(G)| \). In
each phase, we perform a number of iterations, where in each iteration we start with some active cluster \( S' \subseteq S \), and replace it with another cluster \( S'' \subset S' \). Therefore, the number of iterations in each phase is bounded by \( n \).

Canonical Cut: Suppose we are given a collection \( C \) of disjoint vertex subsets in graph \( G \). We say that a cut \((A, B)\) in graph \( G \) is canonical w.r.t. \( C \), iff for each \( C \in C \), either \( C \subseteq A \), or \( C \subseteq B \). We say that it is a non-trivial canonical cut, iff both \( A \) and \( B \) contain at least one cluster in \( C \).

1.4.2 Proof of Lemma 5

If \(|S'| = 1\), then clearly \( S' \) is a critical cluster, so we stop the algorithm and return \( S' \). We assume from now on that \(|S'| > 1\). Our algorithm consists of two steps. In the first step, we try to find a large canonical cluster \( S'' \subseteq S' \) that has the bandwidth property. In the second step, we either find a subset \( E_{S''} \subseteq \text{out}(S'') \) of \( L/4 \) edges, and a set \( \mathcal{P}_{S''} : E_{S''} \rightsquigarrow \eta^* \text{out}(S) \) of paths as required, or compute a new collection \( \pi' \) of good clusters with \( W(\pi') < W(\pi) \).

We now turn to describe the two steps.

Step 1. We construct a partition \( \pi_1 \) of the vertices of \( S' \) into small clusters, as follows.

Start with \( \pi_1 = \{C \in \pi \mid C \subseteq S'\} \). Let \( \tilde{S}' \) be the subset of vertices of \( S' \) that do not belong to any cluster of \( \pi_1 \). If any vertex \( v \in \tilde{S}' \) has a degree greater than \( L \) in graph \( G \), then we set \( S'' = \{v\} \) and proceed to the second step (Clearly, \( S'' \) is a large cluster with the bandwidth property). Assume now that every vertex \( v \in \tilde{S}' \) has degree at most \( L \) in \( G \). We then add every vertex of \( \tilde{S}' \) as a separate cluster to \( \pi_1 \). That is, the set of the new clusters that we add to \( \pi_1 \) is \( \{(v) \mid v \in \tilde{S}'\} \). Observe that \( \pi_1 \) now defines a partition of the vertices of \( S' \) into small clusters that all have the bandwidth property.

Our next step is to establish whether \((S', \pi_1)\) has the weight property. We construct the graph \( H_{S'} \), which is obtained from \( G[S'] \) by contracting every cluster \( C \in \pi_1 \) into a
super-node \( v_C \). We set the weight of \( v_C \) be \( w(v_C) = |\text{out}_G(C)| \). We then run the algorithm \( \mathcal{A}_{\text{ARV}} \) to find an approximate sparsest cut \((A, B)\) in graph \( H_{S'} \).

If \( |E_{H_{S'}}(A, B)| \geq \lambda \cdot \alpha_{\text{ARV}}(n) \cdot \min \{ \sum_{v \in A} w(v), \sum_{v \in B} w(v) \} \), then we are guaranteed that for every partition \((\tilde{A}, \tilde{B})\) of the vertices of \( H_{S'} \),

\[
|E_{H_{S'}}(\tilde{A}, \tilde{B})| \geq \lambda \cdot \min \left\{ \sum_{v \in \tilde{A}} w(v), \sum_{v \in \tilde{B}} w(v) \right\},
\]

and so the weight property holds for the current partition \( \pi_1 \). We then return \( S' \) as a critical cluster, together with the partition \( \pi^* = \pi_1 \).

We assume from now on that \( |E_{H_{S'}}(A, B)| < \lambda \cdot \alpha_{\text{ARV}}(n) \cdot \min \{ \sum_{v \in A} w(v), \sum_{v \in B} w(v) \} \). Let \( A' \) be the subset of vertices obtained from \( A \), after we un-contract all clusters in \( \pi_1 \), and let \( B' \) be the set obtained similarly from \( B \). So \((A', B')\) is a partition of \( S' \). Denote \( T_A = |\text{out}_G(A') \cap \text{out}_G(S')| \), and \( T_B = |\text{out}_G(B') \cap \text{out}_G(S')| \). Recall that there is a one-to-one correspondence between \( E_{H_{S'}}(A, B) \) and \( E_G(A', B') \). We therefore do not distinguish between these two sets of edges and denote both of them by \( \Gamma \). We assume w.l.o.g. that \( |T_A| \leq |T_B| \). As observed before,

\[
|\Gamma| \leq \lambda \cdot \alpha_{\text{ARV}}(n) \cdot \sum_{v \in A} w(v) \tag{1.4}
\]

On the other hand, from the bandwidth property of the cluster \( S' \), we get that \( |\Gamma| \geq \alpha_{\text{BW}} \cdot |T_A| \), and so \( |\text{out}_G(A')| = |\Gamma| + |T_A| \leq \left( 1 + \frac{1}{\alpha_{\text{BW}}} \right) \cdot |\Gamma| \leq \frac{2}{\alpha_{\text{BW}}} |\Gamma| \).

Combining this with equation (1.4), we get that:

\[
\sum_{v \in A} w(v) \geq \frac{|\Gamma|}{\lambda \cdot \alpha_{\text{ARV}}(n)} \geq |\text{out}_G(A')| \frac{\alpha_{\text{BW}}}{2\lambda \cdot \alpha_{\text{ARV}}(n)} \geq 4 |\text{out}_G(A')| \tag{1.5}
\]

since \( \lambda = \frac{\alpha_{\text{BW}}}{8 \alpha_{\text{ARV}}(n)} \).

We now construct a new graph \( H \) from graph \( G \), as follows. First, we sub-divide every edge \( e \in \text{out}_G(A') \) by a vertex \( v_e \), and we let \( T'' = \{ v_e \mid e \in \text{out}_G(A') \} \). We then let \( H \)
be the sub-graph of $G$ induced by $A' \cup T''$, after we contract all clusters $C \in \pi$ that are contained in $A'$. Observe that $H$ is identical to $H_{S'}[A]$, except for the edges adjacent to the vertices in $T''$ that are added in graph $H$. In particular, $V(H) \setminus T' = A$. We perform the extended well-linked decomposition of the set $A$ of vertices in graph $H$, using Theorem 6, obtaining a decomposition $\mathcal{W}'$. We now consider two cases.

**Case 1:** The first case happens if all clusters in $\mathcal{W}'$ are small. In this case, for each cluster $R \in \mathcal{W}'$, let $R'$ be the set of vertices obtained from $R$ after we un-contract all clusters in $\pi$. We run another round of extended well-linked decomposition on each such cluster $R'$, obtaining a partition $\mathcal{W}_{R'}$ of $R'$. We let $\mathcal{W} = \bigcup_{R \in \mathcal{W}'} \mathcal{W}_{R'}$ be the resulting partition of all vertices in $A'$. Observe that we are guaranteed that every cluster in $\mathcal{W}$ is small and has the bandwidth property. We obtain a new good collection $\pi'$ of clusters from $\pi$, by first removing from $\pi$ all clusters $C \subseteq A'$, and then adding all clusters from $\mathcal{W}$. Since

$$\sum_{C \in \pi_1: C \subseteq A'} |\text{out}_G(C)| \geq 4|\text{out}_G(A')|,$$

we are guaranteed that $W(\pi') < W(\pi)$. We then stop the algorithm and return $\pi'$.

**Case 2:** In the second case, there is at least one large cluster $\tilde{S} \in \mathcal{W}'$. Let $S''$ be the subset of vertices obtained from $\tilde{S}$ after we un-contract every cluster $C \in \pi_1$. Then $S'' \subseteq S'$ is a large canonical cluster w.r.t. $\pi$. We claim that $S''$ has the bandwidth property. Assume otherwise, and let $(X,Y)$ be a violating partition of $S''$. Let $T_X = \text{out}(X) \cap \text{out}(S'')$, $T_Y = \text{out}(Y) \cap \text{out}(S'')$, $E' = E_G(X,Y)$, and assume w.l.o.g. that $|T_X| \leq |T_Y|$. Since $(X,Y)$ is a violating partition, $|E'| < \alpha_{bw} \cdot |T_X|$. Let $\pi'' \subseteq \pi$ be the collection of clusters contained in $S''$. We now modify the partition $(X,Y)$ of $S''$, so that for every cluster $C \in \pi''$, either $C \subseteq X$ or $C \subseteq Y$ holds. Consider any such cluster $C \in \pi''$, and let $E_C' = E_G(C \cap X, C \cap Y)$. We partition the edges in $\text{out}_G(C)$ into four subsets, $E_X, E_Y, E_{XY}$, and $E_{YX}$, as follows.
Let \((u, v) \in \text{out}_G(C)\) with \(u \in C\). If \(u, v \in X\), then \(e\) is added to \(E_X\); if \(u, v \in Y\), then \(e\) is added to \(E_Y\); if \(u \in X, v \in Y\), then \(e\) is added to \(E_{XY}\); otherwise it is added to \(E_{YX}\). If \(|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|\), then we move all vertices of \(C\) to \(Y\), and otherwise we move them to \(X\). Assume w.l.o.g. that \(|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|\), so we have moved the vertices of \(C\) to \(Y\). The only new edges added to the cut are the edges of \(E_X\), and since \(C\) is a small cluster with the bandwidth property, \(|E'_C| \geq \alpha_S \cdot |E_X|\). We charge the edges of \(E'_C\) for the edges of \(E_X\), where the charge to every edge of \(E'_C\) is at most \(1/\alpha_S\).

Let \((X', Y')\) be the final partition, obtained after all clusters \(C \in \pi''\) have been processed. Notice that the vertices of \(\mathcal{T}''\) do not belong to any cluster \(C \in \pi''\), so the partition of \(\mathcal{T}''\) induced by \((X', Y')\) is the same as the partition induced by \((X, Y)\). From the above charging scheme, since in the original partition, \(|E_G(X, Y)| < \alpha_{BW} \cdot |T_X|\), in the new partition \(|E_G(X', Y')| \leq \frac{1}{\alpha_S} |E_G(X, Y)| < \alpha_W \cdot |T_X|\), since \(\alpha_{BW} = \alpha_S \cdot \alpha_W\). Finally, notice that partition \((X', Y')\) of \(S''\) naturally defines a partition \((\tilde{X}, \tilde{Y})\) of \(\tilde{S}\), where for each cluster \(C \in \pi''\), \(v_C \in \tilde{X}\) iff \(C \subseteq X'\). But then \(|E_{H}(\tilde{X}, \tilde{Y})| = |E_G(X', Y')| < \alpha_W \min \{ |\text{out}_H(\tilde{X}) \cap \text{out}_H(\tilde{S})|, |\text{out}_H(\tilde{Y}) \cap \text{out}_H(\tilde{S})| \}\), contradicting the fact that \(\tilde{S}\) is \(\alpha_W\)-well-linked in \(H\). We conclude that \(S''\) has the bandwidth property.

**Step 2.** We now assume that we are given a large cluster \(S'' \subseteq S'\) that has the bandwidth property, and \(S''\) is canonical for \(\pi\). The goal of this step is to find a collection \(E_{S''} \subseteq \text{out}_G(S'')\) of \(L/4\) edges, and the set \(\mathcal{P}_{S''}\) of paths as required. If we fail to find such a collection of paths, then we will compute a new good collection \(\pi'\) of clusters, such that \(W(\pi') < W(\pi)\).

We set up the following flow network. Let \(\tilde{S}\) be the set of vertices of \(Z\), obtained from \(S''\) by replacing every cluster \(C \in \pi\) with \(C \subseteq S''\), by the super-node \(v_C\). Start with the current contracted graph \(Z\), and contract all vertices of \(\tilde{S}\) into the source \(s\). Contract all vertices in \(T'\) (recall that these are the vertices \(\{v_e \mid e \in \text{out}(S)\}\)) into the sink \(t\). Let \(N\) be the resulting flow network. We now try to find a flow \(F'\) of value \(L/4\) from \(s\) to \(t\) in
\( \mathcal{N} \). Assume first that such flow can be found. From the integrality of flow, there is a set \( E_{S''} \subseteq \text{out}_G(S'') \) of \( L/4 \) edges, and a set \( \mathcal{P}' \) of \( L/4 \) edge-disjoint paths, where each path connects a distinct edge of \( E_{S''} \) to a distinct vertex of \( T' \) in graph \( Z \), and the paths in \( \mathcal{P}' \) cause congestion 1 in \( Z \). We now show how to find a flow \( F^* : E_{S''} \leadsto \eta^* \text{out}(S) \) in graph \( G \), where each edge in \( \text{out}(S) \) receives at most one flow unit. The flow-paths in \( F^* \) will follow the paths in \( \mathcal{P}' \), except that we need to specify the routing inside the clusters \( C \in \pi \). For each such cluster \( C \), the set \( \mathcal{P}' \) of paths defines a set \( D_C \) of integral 1-restricted demands on the edges of \( \text{out}_G(C) \). Since each such cluster \( C \) is \( \alpha_S \)-well-linked, and it is a small cluster, this set of demands can be routed inside \( C \) with congestion at most \( \frac{2\beta_{\text{FCG}}(L)}{\alpha_S} \leq \eta^* \). From the integrality of flow, we can find a set \( \mathcal{P}^* : E_{S''} \leadsto \eta^* \text{out}(S) \) of paths in \( G[S \setminus S''] \), where each edge in \( \text{out}(S) \) serves as an endpoint of at most one such path. We return \( S'', E_{S''} \) and \( \mathcal{P}^* \) as our output.

Finally, assume that such flow does not exist. From the min-cut/max-flow theorem, we can find a cut \( R \) in graph \( Z \), with \( |\text{out}(R)| < L/4 \), and \( \tilde{S} \subseteq R \). Since \( S'' \) is a large cluster, \( \sum_{v \in R} |\text{out}_Z(v)| \geq L \). Let \( R' \) be the subset of vertices of \( G \) obtained by un-contracting all clusters \( C \in \pi \) that are contained in \( R \). We perform the extended well-linked decomposition of \( R' \), using Theorem 6. Let \( \mathcal{W}'' \) be the resulting decomposition. Since \( |\text{out}_Z(R)| < L/4 \), all clusters in \( \mathcal{W}'' \) are small. We obtain a new collection \( \pi' \) of good clusters from \( \pi \), by first removing all clusters contained in \( R' \), and then adding the clusters in \( \mathcal{W}'' \). Since \( \sum_{C \in \mathcal{W}''} |\text{out}_G(C)| \leq 2|\text{out}_G(R')| < L/2 \), while \( \sum_{v \in R} |\text{out}_Z(v)| \geq |\text{out}_G(S'')| > L \), we are guaranteed that \( W(\pi') < W(\pi) \).

### 1.5 Routing across Small and Critical Clusters

Following the ideas of Andrews [5], we will treat critical clusters as contracted nodes and solve a routing problem in the resulting contracted graph. To be able to do so, we need to ensure that, for any small or critical cluster \( S \), demands between edges in \( \text{out}(S) \) can be routed with low congestion inside \( S \).
The following two theorems allow us to route across these clusters. The proof of Theorem 9 appears in Section 1.5.1.

**Theorem 9** Given any small cluster $S$ that has the bandwidth property, we can efficiently find a partition $G_S$ of the edges of $\text{out}(S)$ into groups of size at most $z = \text{poly log log } n$, such that, for any $\gamma \geq 1$, given any $(\gamma, G_S)$-restricted set $\mathcal{D}$ of demands on the edges of $\text{out}(S)$, there is an efficient randomized algorithm, that w.h.p. finds an integral routing of $\mathcal{D}$ inside $S$ with congestion at most $60\gamma$.

The following theorem gives an efficient algorithm for integral routing across critical clusters.

**Theorem 10** Suppose we are given any cluster $S$, together with a partition $\pi$ of $S$ into small clusters, such that every cluster $C \in \pi$ is $\alpha_S/3$-well-linked, and $(S, \pi)$ has the weight property with parameter $\lambda/3$. Then we can efficiently find a partition $G$ of the edges of $\text{out}(S)$ into groups of size at least $Z = O(\log^4 n)$ and at most $3Z$, such that, for any set $\mathcal{D}$ of $(1, G)$-restricted demands on $\text{out}(S)$, there is an efficient randomized algorithm that w.h.p. routes $\mathcal{D}$ integrally in $G[S]$ with congestion at most $721$.

### 1.5.1 Proof of Theorem 9

We use the following theorem from [40] to route demands across small clusters.

**Theorem 11** Let $G$ be any graph and $\mathcal{T}$ any subset of $k$ vertices called terminals, such that $G$ is $\alpha$-well-linked for $\mathcal{T}$. Then we can efficiently find a partition $G$ of the terminals in $\mathcal{T}$ into groups of size $\frac{\text{poly log } k}{\alpha}$, such that, for any $(1, G)$-restricted set $\mathcal{D}$ of demands on $\mathcal{T}$, there is an efficient randomized algorithm that w.h.p. finds an integral routing of $\mathcal{D}$ in $G$ with edge congestion at most 15.

Suppose we are given a small cluster $S$ that has the bandwidth property. Since $|\text{out}(S)| \leq \text{poly log } n$, and $S$ is $\alpha_S$-well-linked, we can use Theorem 11 to find a partition $G_S$ of the edges
of $\text{out}(S)$ into sets of size $\text{poly log log } n$, such that any $(1, \mathcal{G}_S)$-restricted set $\mathcal{D}$ of demands can be integrally routed inside $S$ with congestion 15 w.h.p.

**Observation 1** Let $\mathcal{G}$ be any partition of the set $\mathcal{T}$ of terminals, and let $\mathcal{D}$ be any set of $(\gamma, \mathcal{G})$-restricted integral demands. Then we can efficiently find $4\gamma$ sets $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$ of $(1, \mathcal{G})$-restricted integral demands, such that any routing of the demands in set $\bigcup_{i=1}^{4\gamma} \mathcal{D}_i$ gives a routing of the demands in $\mathcal{D}$ with the same congestion, and moreover, if the former routing is integral, so is the latter.

**Proof:** Let $\mathcal{G} = \{\mathcal{T}_1, \ldots, \mathcal{T}_r\}$. Our first step is to modify the set $\mathcal{D}$ of demands, so that it does not contain demand pairs that belong to the same set $\mathcal{T}_i$. Specifically, for every pair $(u, v) \in \mathcal{D}$, where $u, v \in \mathcal{T}_i$ for some $1 \leq i \leq r$, we replace the demand $(u, v)$ with a pair of demands $(u, x)$, $(v, x)$, where $x$ is any vertex in set $\mathcal{T}_{i+1}$ (if $i = r$, then $x$ is any vertex in $\mathcal{T}_1$). Let $\mathcal{D}'$ be the resulting set of demands. Clearly, any routing of $\mathcal{D}'$ gives a routing of $\mathcal{D}$ with the same congestion, and if the routing of $\mathcal{D}'$ is integral, so is the corresponding routing of $\mathcal{D}$. It is also easy to see that $\mathcal{D}'$ is $(2\gamma, \mathcal{G})$-restricted.

Our second step is to decompose $\mathcal{D}'$ into $4\gamma$ demand sets $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$, such that each set $\mathcal{D}_j$ of demands is $(1, \mathcal{G})$-restricted, and $\bigcup_{j=1}^{4\gamma} \mathcal{D}_j = \mathcal{D}'$. We construct a multi-graph graph $H$ with vertices $v_1, \ldots, v_r$ corresponding to the groups $\mathcal{T}_1, \ldots, \mathcal{T}_r$ of $\mathcal{G}$. For every pair $(u, v) \in \mathcal{D}'$, with $u \in \mathcal{T}_i$, $v \in \mathcal{T}_j$, we add an edge $(i, j)$ to graph $H$. Finding the decomposition $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$ of the set $\mathcal{D}'$ of demands then amounts to partitioning the edges of $H$ into $4\gamma$ matchings. Since the maximum vertex degree in $H$ is at most $2\gamma$, such a decomposition can be found by a simple greedy algorithm.

Combining Theorem 11 with Observation 1, we get the proof of the theorem.

### 1.5.2 Routing across Critical Clusters

We build the following auxiliary graph $G'$. Start from graph $G$, and subdivide every edge $e \in \text{out}(S)$ by a terminal $t_e$. Let $\mathcal{T}' = \{t_e \mid e \in \text{out}_G(S)\}$. Graph $G'$ is the sub-graph of
$G$ induced by $S \cup T'$. Instead of routing demands over the edges of out($S$), we can now equivalently route pairs of terminals in $T'$, in graph $G'$.

Let $T$ be any spanning tree of $G'$. We use the standard grouping technique to group the terminals in $T'$ along the tree $T$ into groups of size at least $Z$ and at most $3Z$. Let $U_1, \ldots, U_r$ denote these groups. Let $\mathcal{G}'$ be the resulting partition of the terminals in $T'$. This partition defines the final partition $\mathcal{G}$ of the edges in out($S$). Assume now that we are given any $(1, \mathcal{G})$-restricted set $\mathcal{D}$ of demands on the set out($S$) of edges. Let $\mathcal{D}'$ be the corresponding set of demands on the set $T'$ of terminals. It is enough to prove that the demands in $T'$ can be routed in graph $G'$ with congestion at most 721. We assume w.l.o.g. that $r$ is even, and that $\mathcal{D}' = \{(t_1, t_2), (t_3, t_4), \ldots, (t_{r-1}, t_r)\}$, where for each $1 \leq i \leq r$, $t_i \in U_i$.

Our first step is to extend the set $\mathcal{D}'$ of demands as follows. For each $1 \leq i \leq r$, let $U'_i \subseteq U_i$ be any subset of exactly $Z$ terminals. For each $1 \leq j \leq r/2$, let $M_j$ be any complete matching between $U'_{2j-1}$ and $U'_{2j}$. The new set $\mathcal{D}^+$ of demands is $\mathcal{D}^+ = \bigcup_{j=1}^{r/2} M_j$.

Let $\pi$ be the partition of $S$ into small sub-clusters that have the bandwidth property, such that $(S, \pi)$ has the weight property. We define another auxiliary graph $H'$, as follows. Start from graph $G'$, and contract every cluster $C \in \pi$ into a super-node $v_C$. The rest of the proof consists of two steps. First, we show that the set $\mathcal{D}^+$ of demands can be fractionally routed with small congestion on short paths in graph $H'$. Next, we use the Lovasz Local Lemma to transform this fractional routing into an integral routing of the set $\mathcal{D}'$ of demands in graph $G'$.

**Step 1: Routing on short paths in $H'$**

**Definition 4** We say that a graph $G = (V, E)$ is an $\alpha$-expander, iff

$$\min_{X:|X| \leq |V|/2} \left\{ \frac{|E(X, \overline{X})|}{|X|} \right\} \geq \alpha$$

We will use the result of Leighton and Rao [71], who show that any demand that is
routable on an expander graph with no congestion, can also be routed on relatively short paths with small congestion. In order to use their result, we need to turn $H'$ into a constant-degree expander. We do so as follows.

Recall that the vertices of the graph $H'$ are of two types: terminals of $T'$, and super-nodes $v_C$ for $C \in \pi$. Moreover, from the weight property, for any partition $(A, B)$ of $V(H')$,

$$|E_{H'}(A, B)| \geq \lambda \cdot \min \left\{ \sum_{v_C \in A: C \in \pi} d_{H'}(v_C), \sum_{v_C \in B: C \in \pi} d_{H'}(v_C) \right\}$$

We process the vertices $v_C$ of $H'$ that correspond to the super-nodes one-by-one. Let $v$ be any such vertex, let $d$ be its degree, and let $e_1, \ldots, e_d$ be the edges adjacent to $v$. We replace $v$ with a degree-3 expander $X_v$ on $d$ vertices, whose expansion parameter is some constant $\alpha'$. Each edge $e_1, \ldots, e_d$ now connects to a distinct vertex of $X_v$. Let $H''$ denote the graph obtained after each super-node of $H'$ has been processed. Notice that the maximum vertex degree in $H''$ is bounded by 4, $T' \subseteq V(H'')$, and any fractional routing of the set $D^+$ of demands in graph $H''$ on paths of length at most $\ell$ gives a routing of the same set of demands in $H'$, with the same congestion, on paths of length at most $\ell$. We next show that graph $H''$ is an $\alpha$-expander, for $\alpha = \lambda \alpha'/4$.

**Claim 2** Graph $H''$ is an $\alpha$-expander, for $\alpha = \lambda \alpha'/4$.

**Proof:** Assume otherwise, and let $(A, B)$ be a violating cut, that is, $|E_{H''}(A, B)| < \alpha \cdot \min \{|A|, |B|\}$. Notice that for each terminal $t \in T'$, there is exactly one vertex $v_t \in V(H'')$ adjacent to $t$, and we can assume w.l.o.g. that both $t$ and $v_t$ belong to the same set, $A$ or $B$ (otherwise $t$ can be moved to the other set, and the sparsity of the cut will only go down).

We use the cut $(A, B)$ to define a partition $(A', B')$, where $A', B' \neq \emptyset$, of the vertices of $H'$, and show that $|E_{H'}(A', B')| < \lambda \cdot \min \left\{ \sum_{v_C \in A': C \in \pi} d_{H'}(v_C), \sum_{v_C \in B': C \in \pi} d_{H'}(v_C) \right\}$, thus contradicting the weight property of $(S, \pi)$.

Partition $(A', B')$ is defined as follows. For each terminal $t \in T'$, if $t \in A$, then we add
to $A'$; otherwise it is added to $B'$. For each super-node $v_C$, if at least half the vertices of $X_{v_C}$ belong to $A$, then we add $v_C$ to $A'$; otherwise we add $v_C$ to $B'$.

We claim that $|E_{H'}(A', B')| \leq |E_{H''}(A, B)|/\alpha'$. Indeed, consider any super-node $v_C \in V(H')$, and consider the partition $(A_{v_C}, B_{v_C})$ of the vertices of $X_{v_C}$ defined by the partition $(A, B)$, that is, $A_{v_C} = A \cap V(X_{v_C}), B_{v_C} = B \cap V(X_{v_C})$. Assume w.l.o.g. that $|A_{v_C}| \leq |B_{v_C}|$. Then the contribution of the edges of $X_{v_C}$ to $E_{H''}(A, B)$ is at least $\alpha' \cdot |A_{v_C}|$. After vertex $v_C$ is processed, we add at most $|A_{v_C}|$ edges to the cut. Therefore,

$$|E_{H'}(A', B')| \leq \frac{|E_{H''}(A, B)|}{\alpha'} \leq \frac{\alpha'}{\alpha'} \cdot \min \{|A|, |B|\} = \frac{\lambda}{4} \min \{|A|, |B|\}$$

Assume w.l.o.g. that $\sum_{v_C \in A'} d_{H'}(v_C) \leq \sum_{v_C \in B'} d_{H'}(v_C)$. Consider the set $A$ of vertices of $H''$, and let $A_1 \subseteq A$ be the subset of vertices, that belong to expanders $X_{v_C}$, where $|V(X_{v_C}) \cap A| \leq |V(X_{v_C}) \cap B|$. Notice that from the expansion properties of graphs $X_{v_C}$, $|E_{H''}(A, B)| \geq \alpha'|A_1|$, and so $|A_1| \leq \frac{|E_{H''}(A, B)|}{\alpha'} \leq \frac{\alpha'}{\alpha'} |A| \leq \frac{|A|}{8}$. Each non-terminal vertex in $A \setminus A_1$ contributes at least 1 to the summation $\sum_{v_C \in A'} d_{H'}(v_C)$, and for each terminal $t \in T'$, its unique neighbor belongs to $A$, so $|T' \cap A| \leq |A|/2$, and $|A \setminus (A_1 \cup T')| \geq \frac{3}{8}|A|$. Therefore, $\sum_{v_C \in A'} d_{H'}(v_C) \geq \frac{3}{8}|A|$. We conclude that:

$$|E_{H'}(A', B')| \leq \frac{\lambda}{4} |A| \leq \frac{2\lambda}{3} \sum_{v_C \in A'} d_{H'}(v_C)$$

contradicting the weight property of $(S, \pi)$.

The following theorem easily follows from the results of Leighton and Rao [71], and its proof can be found in [40].

**Theorem 12** Let $G$ be any $n$-vertex $\alpha$-expander with maximum vertex degree $D_{max}$, and let $M$ be any partial matching over the vertices of $G$. Then there is an efficient randomized algorithm that finds, for every pair $(u, v) \in M$, a collection $\mathcal{P}_{u,v}$ of $m = \Theta(\log n)$ paths of length $O(D_{max} \log n/\alpha)$ each, such that the set $\mathcal{P} = \bigcup_{(u, v) \in M} \mathcal{P}_{u,v}$ of paths causes congestion $O(\log^2 n/\alpha)$ in $G$. The algorithm succeeds with high probability.
We now apply Theorem 12 to the set $D^+$ of demands in graph $H''$. For every pair $(u, v)$ of terminals with $D^+(u, v) = 1$, we obtain a set $\mathcal{P}'_{u,v}$ of $m = O(\log n)$ paths in graph $H''$, of length at most $\ell = O(\log n/\lambda) = O(\log^3 n \text{poly} \log \log n)$ each, and the paths in $\bigcup_{(u,v):D^+(u,v)=1} \mathcal{P}'_{u,v}$ cause congestion at most $\tilde{\eta} = O(\log^2 n/\lambda) = O(\log^4 n \text{poly} \log \log n)$ in graph $H''$. As observed before, for each pair $(u, v)$ of terminals with $D^+(u, v) = 1$, the set $\mathcal{P}'_{u,v}$ gives a set $\mathcal{P}_{u,v}$ of paths in graph $H'$, connecting $u$ to $v$, such that the length of each path in $\mathcal{P}_{u,v}$ is at most $\ell$. Let $\mathcal{P} = \bigcup_{(u,v):D^+(u,v)=1} \mathcal{P}_{u,v}$. Then the paths in $\mathcal{P}$ cause congestion at most $\tilde{\eta}$ in $H'$. This concludes the first step of the algorithm.

**Step 2: Integral routing in $G'$** For each $1 \leq j \leq r/2$, let $\mathcal{B}_j$ be the union of the sets of paths $\mathcal{P}_{u,v} \subseteq \mathcal{P}$ where $u \in U_{2j-1}, v \in U_{2j}$. We call the set $\mathcal{B}_j$ of paths a **bundle**.

Let $c = 13$. We set $Z = \frac{2\tilde{\eta}^{1+\varepsilon^{-1}}z^{1+\varepsilon^{-1}}l^{1+\varepsilon^{-1}}}{m} = O((\log n)^{3+\frac{11}{c-1}} \cdot \text{poly} \log \log n) = O(\log^4 n)$, where $z$ is the parameter from Corollary 9.

For each small cluster $C \in \pi$, let $G_C$ be the partition of the edges of $\text{out}_G(C)$ guaranteed by Corollary 9. We will select one path from each bundle $\mathcal{B}_j$, such that for each small cluster $C \in \pi$, for each group $U \in G_C$, at most $c$ of the selected paths contain edges of $U$. We do so, using the constructive version of the Lovasz Local Lemma by Moser and Tardos [75]. The next theorem summarizes the symmetric version of the result of [75].

**Theorem 13 ([75])** Let $X$ be a finite set of mutually independent random variables in some probability space. Let $\mathcal{A}$ be a finite set of bad events determined by these variables. For each event $A \in \mathcal{A}$, let $\text{vbl}(A) \subseteq X$ be the unique minimal subset of variables determining $A$, and let $\Gamma(A) \subseteq \mathcal{A}$ be a subset of bad events $B$, such that $A \neq B$, but $\text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset$. Assume further that for each $A \in \mathcal{A}$, $|\Gamma(A)| \leq D$, $\Pr[A] \leq p$, and $ep(D+1) \leq 1$. Then there is an efficient randomized algorithm that w.h.p. finds an assignment to the variables of $X$, such that none of the events in $\mathcal{A}$ holds.

For each bundle $\mathcal{B}_j$, we randomly choose one of the paths $P_j \in \mathcal{B}_j$. Let $x_j$ be the random variable indicating which path has been chosen from $\mathcal{B}_j$. 41
For each small cluster $C \in \pi$, for each group $U \in \mathcal{G}_C$, we define a bad event $\beta_{C,U}$ to be the event that at least $c$ of the chosen paths contain edges in $U$. The set $vbl(\beta_{C,U})$ contains all variables $x_j$, where at least one path in bundle $B_j$ contains an edge of $U$. Since the set $\mathcal{P}$ of paths causes congestion at most $\tilde{\eta}$ in $H'$, and each group contains at most $z$ edges, $|vbl(\beta_{C,U})| \leq \tilde{\eta}z$. The number of potential $c$-tuples of paths (where we take at most one path from each bundle) containing edges of $U$ is at most $\tilde{\eta}^c z$, and each such $c$-tuple is selected with probability at most $1/(Zm)^c$. Therefore, $\Pr[\beta_{C,U}] \leq \left(\frac{\tilde{\eta}z}{Zm}\right)^c$. We denote this probability by $p$.

For each bundle $B_j$, there are $mZ$ paths in the bundle, each path contains $\ell$ edges, and for each such edge $e$, there are most two bad events $\beta_{C,U}, \beta_{C',U'}$, where $e \in U$ and $e \in U'$. Therefore, for each bad event $\beta_{C,U}$, $|\Gamma(\beta_{C,U})| \leq |vbl(\beta_{C,U})| \cdot 2mZ\ell \leq 2mZ\ell\tilde{\eta}z$. We denote $D = 2mZ\ell\tilde{\eta}z$.

It now only remains to verify that $e(D + 1) \cdot p \leq 1$, or equivalently, $e(2mZ\ell\tilde{\eta}z + 1) \cdot \left(\frac{\tilde{\eta}z}{Zm}\right)^c \leq 1$. This is immediate from the choice of $Z = \frac{2\eta^{1+\frac{2}{c-1}} z^{1+\frac{2}{c-1}} \ell^{\frac{1}{c-1}}}{m}$.

Assume now that we have selected a collection $P_1, \ldots, P_{r/2}$ of paths, such that none of the bad events $\beta_{C,U}$ happens. Then for each small cluster $C \in \pi$, the set $P_1, \ldots, P_{r/2}$ of paths define a set $D_C$ of $(c-1, G_C)$-restricted demands, which can be routed inside $C$ with congestion at most $60(c-1) = 720$ using Corollary 9. Let $\mathcal{P}' = \left\{P'_1, \ldots, P'_{r/2}\right\}$ denote the resulting set of paths in graph $G'$. Then for each $1 \leq j \leq r/2$, path $P'_j$ connects some terminal $t'_{2j-1} \in U_{2j-1}$ to some terminal $t'_{2j} \in U_{2j}$. Moreover, since every edge of $H'$ belongs to some group $U \in \mathcal{G}_C$ of some small cluster $C \in \pi$, the total congestion caused by paths in $\mathcal{P}'$ is bounded by 720. Finally, we extend each path $P'_j$ by connecting $t_{2j-1}$ to $t'_{2j-1}$, and $t'_{2j}$ to $t_{2j}$ via the spanning tree $T$. Since each group $U \in \mathcal{G}'$ is associated with a sub-tree $T_U$ of $T$, and all these sub-trees are edge-disjoint, the resulting set of paths gives an integral routing of the set $D'$ of demands in graph $G'$, with congestion at most 721. This concludes the proof of Theorem 10.
1.6 Organization and List of Parameters

We start by presenting our main technical tool in Chapter 2. In the next two chapters, we show how this technical theorem can be applied to the Integral Concurrent Flow problems: We show an algorithm and hardness result for basic-ICF in Chapter 3 and an algorithm and hardness result for group-ICF in Chapter 4. We present these algorithms in the setting where each edge does not have capacity. We argue in Section 5.1 that our algorithmic results can be extended to capacitated problems.

We provide a list of important parameters in this section.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{ARV}(n)$</td>
<td>$O(\sqrt{\log n})$</td>
<td>Approximation factor of $A_{ARV}$ for sparsest cut</td>
</tr>
<tr>
<td>$\beta_{FCG}(k)$</td>
<td>$O(\log k)$</td>
<td>Flow-cut gap for undirected graphs</td>
</tr>
<tr>
<td>$L$</td>
<td>$\text{poly log } n$</td>
<td>Threshold for defining small and large clusters</td>
</tr>
<tr>
<td>$\alpha_W$</td>
<td>$\Omega\left(\frac{1}{\log^{1.5} n}\right)$</td>
<td>Well-linkedness for large clusters by Theorem 6</td>
</tr>
<tr>
<td>$\alpha_S$</td>
<td>$\Omega\left(\frac{1}{(\log \log n)^{1.5}}\right)$</td>
<td>Bandwidth condition of small clusters</td>
</tr>
<tr>
<td>$\alpha_{BW}$</td>
<td>$\Omega\left(\frac{1}{(\log n \log \log n)^{1.5}}\right)$</td>
<td>Bandwidth condition of large clusters</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$\Omega\left(\frac{1}{\log^2 n \cdot (\log \log n)^{1.5}}\right)$</td>
<td>Parameter for weight condition</td>
</tr>
<tr>
<td>$\eta^*$</td>
<td>$O\left((\log \log n)^{2.5}\right)$</td>
<td>A large cut contains more than $L/(4\eta^*)$ edges</td>
</tr>
<tr>
<td>$\alpha_{EDP}$</td>
<td>$\text{poly log } k$</td>
<td>EDP approximation factor from Theorem 5</td>
</tr>
<tr>
<td>$\eta_{EDP}$</td>
<td>14</td>
<td>Congestion from Theorem 5</td>
</tr>
<tr>
<td>$\alpha^*$</td>
<td>$O\left(\log^4 n \text{ poly log } n\right)$</td>
<td>A factor up to which the cuts are preserved in graph $H$ in Theorem 15</td>
</tr>
<tr>
<td>$\alpha_{RZ}$</td>
<td>$O(\log^{10} n)$</td>
<td>Approximation factor from [79] for EDP</td>
</tr>
<tr>
<td>$L_{RZ}$</td>
<td>$\Omega(\log^5 n)$</td>
<td>Requirement on the size of minimum global cut in the algorithm of [79]</td>
</tr>
</tbody>
</table>
Additional parameters for the algorithm for group-ICF.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$O(\log^{25} n)$</td>
<td>Threshold for defining small and large clusters</td>
</tr>
<tr>
<td>$m$</td>
<td>$O(\log n)$</td>
<td>Parameter for canonical instances</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>$O(k \text{ poly log } n)$</td>
<td>We require that $D \geq 640\Delta m \alpha_{\text{edp}} \log^2 n = k \text{ poly log } n$</td>
</tr>
</tbody>
</table>
CHAPTER 2
A GRAPH SPLITTING THEOREM

In this chapter we present the main technical tools that our algorithms use: the $Q$-$J$ decomposition, the construction of the graph $H$, and the splitting of $H$ into sub-graphs. We start with the $Q$-$J$ decomposition.

2.1 Graph Decomposition

We assume that we are given a non-empty collection $Q_0$ of disjoint critical clusters in graph $G$, with the following property: if $(A, B)$ is any non-trivial canonical cut in graph $G$ w.r.t. $Q_0$, then it is a large cut. For motivation, consider the basic-ICF problem, and assume that every cut separating the terminals in $\mathcal{T}$ is a large cut. Then we can set $Q_0 = \{\{t\} \mid t \in \mathcal{T}\}$.

For the group-ICF problem, we will compute $Q_0$ differently, by setting $Q_0 = \{Q\}$ where $Q$ is an arbitrary critical cluster in $G$.

Suppose we are given a collection $\mathcal{Q}$ of disjoint critical clusters, and a collection $\mathcal{J}$ of disjoint small clusters, such that $\mathcal{Q} \cup \mathcal{J}$ is a partition of $V(G)$. Let $E^\mathcal{Q} = \bigcup_{Q \in \mathcal{Q}} \text{out}(Q)$, and let $E^\mathcal{J} = (\bigcup_{J \in \mathcal{J}} \text{out}(J)) \setminus E^\mathcal{Q}$. We say that $(\mathcal{Q}, \mathcal{J})$ is a valid $Q$-$J$ decomposition, iff $Q_0 \subseteq \mathcal{Q}$, and:

P1. Every cluster $J \in \mathcal{J}$ is a small cluster with the bandwidth property, and every cluster $Q \in \mathcal{Q}$ is a critical cluster.

P2. There is a set $N = \{\tau_e \mid e \in E^\mathcal{J}\}$ of paths, called tendrils, where path $\tau_e$ connects $e$ to some edge in $E^\mathcal{Q}$, each edge in $E^\mathcal{Q}$ is an endpoint of at most one tendril, and the total congestion caused by $N$ is at most 3. Moreover, the tendrils do not use edges $e = (u, v)$ where both $u$ and $v$ belong to clusters in $\mathcal{Q}$.

P3. If $(S, \overline{S})$ is any cut in graph $G$, which is non-trivial and canonical w.r.t. $\mathcal{Q}$, then it is a large cut.
We refer to the clusters in $Q$ as the $Q$-clusters, and to the clusters in $J$ as the $J$-clusters. We note that Andrews [5] has (implicitly) defined the $Q$-$J$ decomposition, and suggested an algorithm for constructing it, by using the graph decomposition of Räcke [77] as a black-box. The Räcke decomposition however gives very strong properties - stronger than one needs to construct a $Q$-$J$ decomposition. We obtain a $Q$-$J$ decomposition with slightly better parameters by performing the decomposition directly, instead of using the Räcke decomposition as a black-box. For example, the tendrils in $N$ only cause a constant congestion in our decomposition, instead of a logarithmic one, the well-linkedness of the $J$-clusters is $\text{poly log log } n$ instead of $\text{poly log } n$, and we obtain a better relationship between the parameter $L$ and the size of the minimum cut separating the $Q$-clusters. The algorithm for finding a $Q$-$J$ decomposition is summarized in the next theorem.

**Theorem 14** There is an efficient algorithm, that, given any graph $G$, and a set $Q_0$ of disjoint critical clusters, such that any non-trivial canonical cut w.r.t. $Q_0$ in $G$ is large, produces a valid $Q$-$J$ decomposition of $G$.

**Proof:** For each edge $e \in E_J$, if $v$ is the endpoint of the tendril $\tau(e)$ that belongs to some $Q$-cluster in $Q$, then we say that $v$ is the head of the tendril $\tau(e)$. We also set $Q^* = \bigcup_{Q \in Q} Q$.

We build the clusters in $Q$ gradually. The algorithm performs a number of iterations. We start with $Q = Q_0$, and in each iteration, we add a new critical cluster $Q$ to $Q$. In the last iteration, we produce the set $J$ of the $J$-clusters, and their tendrils, as required.

We now proceed to describe an iteration. Let $Q$ be the set of current $Q$-clusters, and let $Q^* = \bigcup_{Q \in Q} Q$. Let $S_0 = V \setminus Q^*$.

We start by performing the extended well-linked decomposition of the set $S_0$ of vertices, using Theorem 6. Let $W$ be the resulting decomposition, and $N$ the corresponding set of tendrils. If all sets in $W$ are small, then we set $J = W$, and finish the algorithm, so the current iteration becomes the last one. The output is $(Q, J)$, and the final set of tendrils is $N$. We will later show that it has all required properties. Assume now that $W$ contains at
least one large cluster, and denote it by $S$. Let $N_S$ be the set of tendrils originating at edges of $\text{out}(S)$.

We use Lemma 4 to find a critical sub-cluster $Q \subseteq S$, together with the subset $E_Q \subseteq \text{out}(Q)$ of $L/4$ edges, and the set $P_S : E_Q \rightsquigarrow \eta^* \text{out}(S)$ of paths, that are contained in $S \setminus Q$. Let $P'_S : E_Q \rightsquigarrow \eta^* \text{out}(S_0)$ be a collection of paths obtained by concatenating the paths in $P_S$ with the set $N_S$ of tendrils originating at the edges of $\text{out}(S)$. Notice that each edge of $\text{out}(S_0)$ serves as an endpoint of at most one such path, and $|P'_S| = L/4$. We then add $Q$ to $Q$, and continue to the next iteration. This concludes the description of an iteration.

Consider the final collections $Q, J$ of clusters produced by the algorithm. It is immediate to see that Properties (P1)–(P2) hold for it. We only need to establish Property (P3).

Consider any cut $(S, \overline{S})$ in graph $G$, such that for each cluster $Q \in Q$, either $Q \subseteq S$, or $Q \subseteq \overline{S}$, and assume that both $S$ and $\overline{S}$ contain at least one cluster in $Q$. We say that the vertices of $S$ are red and the vertices of $\overline{S}$ are blue.

If both $S$ and $\overline{S}$ contain clusters from $Q_0$, then the cut $(S, \overline{S})$ must be large by our initial assumption. Assume w.l.o.g. that all clusters in $Q_0$ are red. Let $Q$ be the first cluster that has been added to $Q$ over the course of the algorithm, whose vertices are blue. Recall that we have a set $P'_Q$ of $L/4$ paths connecting the edges of $\text{out}(Q)$ to the edges of $\text{out}(S_0)$ with congestion at most $\eta^*$. Therefore, there must be at least $\frac{L}{4\eta^*}$ edges in the cut, so $(S, \overline{S})$ is a large cut. This concludes the proof of Theorem 14.

Given a valid $Q$-$J$ decomposition, it is not hard to construct a graph $H$ with the desired properties. The following theorem mostly follows the construction of [5], with some minor changes.

**Theorem 15** Suppose we are given a valid $Q$-$J$ decomposition $(Q, J)$ for graph $G$. Then there is an efficient randomized algorithm to construct a graph $H$ with $V(H) = \{v_C \mid C \in Q\}$, and for each edge $e = (v_{C_1}, v_{C_2}) \in E(H)$, define a path $P_e$ in graph $G$, connecting some vertex of $C_1$ to some vertex of $C_2$, such that for some value $\alpha^* = O(\log^4 n \text{poly log log } n)$, the following properties hold w.h.p. for graph $H$: 48
C1. For every cut \((A, B)\) in graph \(H\), there is a cut \((A', B')\) in graph \(G\), such that for each \(Q \in \mathcal{Q}\), if \(v_Q \in A\) then \(Q \subseteq A'\), and if \(v_Q \in B\) then \(Q \subseteq B'\), and \(|E_G(A', B')| \leq \alpha^* \cdot |E_H(A, B)|\).

C2. The value of the minimum cut in \(H\) is at least \(\frac{L}{\alpha^*}\).

C3. The paths in set \(\mathcal{P}_H^E = \{ P_e \mid e \in E(H) \}\) cause a constant congestion in graph \(G\).

C4. For each critical cluster \(C \in \mathcal{Q}\), let \(\mathcal{G}_C\) be the grouping of the edges of \(\text{out}(C)\) given by Theorem 10. Then for each group \(U \in \mathcal{G}_C\), at most two paths in \(\mathcal{P}_H^E\) contain an edge of \(U\) as their first or last edge.

**Graph Splitting**

Once we compute the graph \(H\), we can split it into graphs \(H_1, \ldots, H_x\), as follows. For each \(1 \leq j \leq x\), the set of vertices \(V(H_j) = V(H)\). The sets of edges \(E(H_1), \ldots, E(H_x)\) are constructed as follows. Each edge \(e \in E(H)\) independently chooses an index \(j \in \{1, \ldots, x\}\) uniformly at random. Edge \(e\) is then added to graph \(H_j\), where \(j\) is the index chosen by \(e\).

We use the following theorem (a re-statement of Theorem 2.1 from [63]).

**Theorem 16 ([63])** Let \(G = (V, E)\) be any \(n\)-vertex graph with minimum cut value \(C\). Assume that we obtain a sub-graph \(G' = (V, E')\), by adding every edge \(e \in E\) with probability \(p\) to \(E'\), and assume further that \(C \cdot p > 48 \ln n\). Then with probability at least \(1 - O(1/n^2)\), for every partition \((A, B)\) of \(V\), \(|E_{G'}(A, B)| \geq p|E_G(A, B)|\).

Therefore, if we select \(L\) so that \(\frac{L}{x \alpha^*} > 48 \ln n\), then we can perform the graph splitting as described above, and from Theorem 16, for each \(1 \leq j \leq x\), for each partition \((A, B)\) of \(V(H)\), \(|E_{H_j}(A, B)| \geq \frac{|E_H(A, B)|}{2x}\) w.h.p.
2.2 Proof of Theorem 15

Our first step is to construct a joined tendril graph $H^*$, whose construction is identical to the one given by Andrews [5], except that we use the new $Q$-$J$ decomposition. The final graph $H$ is obtained by appropriately sampling the edges of $H^*$.

2.2.1 Joined Tendril Graph

Recall that we are given a set $Q$ of $Q$-clusters, and a set $J$ of $J$-clusters. We have defined $Q^* = \bigcup_{Q \in Q} Q$, $E^Q = \bigcup_{Q \in Q} \text{out}(Q)$, and $E^J = (\bigcup_{J \in J} \text{out}(J)) \setminus E^Q$. Recall that we are also given a set $N = \{\tau_e \mid e \in E^J\}$ of tendrils where path $\tau_e$ connects $e$ to some edge in $E^Q$, each edge in $E^Q$ is an endpoint of at most one tendril, and the total congestion caused by the set $N$ of tendrils is at most 3. Moreover, the tendrils do not use edges whose both endpoints belong to $Q^*$. We define $E' = \bigcup_{J \in J} \text{out}(J)$, that is, $E'$ consists of edges in $E^J$ and $\text{out}(Q^*)$.

We now extend the set $N$ of tendrils to include tendrils $\tau(e)$ for edges $e \in \text{out}(Q^*)$, where for each such edge, $\tau(e) = (e)$. The resulting set $N$ contains a tendril $\tau(e)$ for each edge $e \in E'$, the total congestion due to the set $N$ of tendrils is at most 3, and each edge in $\text{out}(Q^*)$ serves as an endpoint of at most two such tendrils. We are now ready to define the graph $H^*$.

The vertices of graph $H^*$ correspond to the $Q$-clusters, $V(H^*) = \{v_C \mid C \in Q\}$. The set of the edges of $H^*$ consists of two subsets, $E^0$ and $E^1$. For each edge $e = (u, u')$ in $G$, where $u \in C$, $u' \in C'$ for a pair $C, C' \in Q$ of distinct $Q$-clusters, we add the edge $(v_C, v_{C'})$ to $E^0$. In order to define the set $E^1$ of edges, we consider the set $J$ of $J$-clusters. For each $J$-cluster $C \in J$, we define a set $E^C$ of edges, and we eventually set $E^1 = \bigcup_{C \in J} E^C$. The final set of edges of $H^*$ is $E(H^*) = E^0 \cup E^1$.

Consider some $J$-cluster $C \in J$. We now define $E^C$ as follows. Let $X_C$ be a degree-3 $\alpha'$-expander on $|\text{out}(C)|$ vertices, where $\alpha'$ is some constant, and let $f_C : V(X_C) \to \text{out}(C)$ be an arbitrary bijection, mapping each vertex of $X_C$ to an edge of $\text{out}(C)$. For each edge
Consider the edges $e_1 = f_C(u)$, and $e_2 = f_C(v)$, and their tendrils $\tau(e_1), \tau(e_2)$. Let $h_1, h_2 \in Q^*$ be the heads of these two tendrils, respectively, and assume that $h_1 \in C_1, h_2 \in C_2$, where $C_1, C_2 \in \mathcal{Q}$. If $C_1 \neq C_2$, then we add an edge $e'$, connecting $v_{C_1}$ to $v_{C_2}$, to set $E^C$. The following notation will be useful later. We denote $p_1(e') = e_1$, and $p_2(e') = e_2$ (the ordering of these two vertices is chosen arbitrarily). We also denote by $\tau_1(e') = \tau(e_1), \tau_2(e') = \tau(e_2)$, and refer to them as the first and the second tendrils of $e'$, respectively. Finally, we denote $f'(e') = e$. Observe that each edge $e'' \in \text{out}(C)$ may serve as $p_1(e')$ or $p_2(e')$ for at most three edges $e' \in E^C$, since the degree of the expander $X_C$ is 3. Notice also that for each edge $e \in E(X_C)$, we add at most one edge $e'$ to $E^C$. We set $E^1 = \bigcup_{C \in J} E^C$, and the final set of edges of $H^*$ is $E(H^*) = E^0 \cup E^1$. For consistency, for edges $e \in E^0$, we define the first and the second tendril of $e$ to be the edge $e$ itself. Let $e \in \bigcup_{J \in \mathcal{J}} \text{out}(J)$, and let $\tau(e)$ be its tendril. Assume that $e = (u, v)$, where $u \in C_1, v \in C_2$. Notice that $\tau(e)$ may serve as the first or the second tendril of at most six edges in graph $H^*$: three edges in $E^{C_1}$ (edges $e'$ for which $p_1(e') = e$, or $p_2(e') = e$), and three edges in $E^{C_2}$. This completes the definition of graph $H^*$.

In the next theorem, we show that the cuts in graph $H^*$ are roughly at least as high as their counterparts in graph $G$. The theorem and its proof are identical to the ones in [5]. We include the proof here for completeness.

**Theorem 17** Let $(A, B)$ be any cut in graph $H^*$. Then there is a cut $(A', B')$ in graph $G$, such that for each $Q \in \mathcal{Q}$, if $v_Q \in A$, then $Q \subseteq A'$, and if $v_Q \in B$, then $Q \subseteq B'$. Moreover, $|E_G(A', B')| \leq O(|E_{H^*}(A, B)|)$.

**Proof:** We will call the vertices of $A$ and $B$ in $H^*$ red and blue, respectively. We will assign the colors, red or blue, to the vertices of $G$, and then the cut $(A', B')$ will be defined based on the colors of vertices, where $A', B'$ contain vertices whose colors are red and blue respectively.

For each critical cluster $Q \in \mathcal{Q}$, if $v_Q$ is red, then we color all its vertices red, and otherwise
we color all its vertices blue. Now, consider some $J$-cluster $J \in \mathcal{J}$. Recall that each edge $e \in \text{out}(J)$ has a tendril $\tau(e)$ associated with it, that connects $e$ to some vertex in $Q^*$. This vertex is already colored red or blue. If at least half the tendrils in set $\{\tau(e) \mid e \in \text{out}(J)\}$ have their head colored red, then we color all vertices of $J$ red; otherwise we color all vertices of $J$ blue. This completes the definition of the cut $(A', B')$ of $V(G)$.

We now show that $|E_G(A', B')| \leq O(|E_{H^*}(A, B)|)$, by defining a mapping from $E_G(A', B')$ to $E_{H^*}(A, B)$, where the number of edges mapped to each edge of $E_{H^*}(A, B)$ is bounded by a constant. We say that an edge of graph $H^*$ or of graph $G$ is multi-colored iff one of its endpoints is red and the other is blue. Consider some multi-colored edge $e$ in graph $G$. Since the endpoints of this edge have different colors, it connects vertices from two different clusters, $C$ and $C'$. If both clusters are critical, then we map $e$ to itself. We will ignore such edges from now on, and we let $\hat{E}$ denote the remaining multi-colored edges. If exactly one of the two clusters (say $C$) is a $J$-cluster, and the other one is a critical cluster, then we say that cluster $C$ is responsible for the edge $e$. Assume now that both clusters $C, C'$ are $J$-clusters, where $C$ is red and $C'$ is blue. If the head of $\tau(e)$ is red, then we say that $C'$ is responsible for $e$; otherwise we say that $C$ is responsible for $e$.

So far we have identified, for each edge $e \in \hat{E}$, a cluster $C(e)$ that is responsible for $e$. This cluster has the property that its color is opposite of the color of the head of $\tau(e)$. Consider any $J$-cluster $C$, and assume that $C$ is responsible for $n_C$ edges. Recall that we have built an expander $X_C$, and a bijection $f_C : V(X_C) \to \text{out}(C)$. We now show that the number of multi-colored edges in $E^C$ is at least $\Omega(n_C)$.

We color the vertices of the expander $X_C$ as follows. Recall that each vertex $v \in V(X_C)$ corresponds to an edge $e = f(v) \in \text{out}(C)$. The edge $e$ is in turn is connected to $Q$ with a tendril $\tau(e)$, whose head is colored either red or blue. If it is colored red, then we color $v$ red as well, and if it is colored blue, then we color $v$ blue.

Assume w.l.o.g. that $C$ is colored blue. Then for each one of the $n_C$ edges $e$ for which $C$ is responsible, the head of $\tau(e)$ is red, while the majority tendrils in set $\{\tau(e) \mid e \in \text{out}(C)\}$
have blue endpoints (that is why we have colored $C$ blue). So the red-blue coloring of $X_C$ must define a cut of size at least $\Omega(n_C)$ in graph $X_C$. If an edge $e \in E(X_C)$ belongs to this cut, then its corresponding edge $e' \in E^C$ must be a multi-colored edge. So the number of multi-colored edges in $E^C$ is at least $\Omega(n_C)$. Overall, the number of multi-colored edges in $E^1$ is at least $\Omega(|\hat{E}|)$, and the total number of multi-colored edges in $H^*$, $|E_{H^*}(A,B)| \geq \Omega(|E_G(A',B')|)$ as required. 

Combining Theorem 17 with (Property P3) of the $Q$-$J$ decomposition, we obtain the following corollary.

**Corollary 3** The value of the minimum cut in graph $H^*$ is $\Omega\left(\frac{L}{n^r}\right)$.

### 2.2.2 Graph $H$

We now complete the construction of graph $H$, with Properties (C1)–(C4). Recall that graph $H^*$ already has the first two properties. The idea is that we will suitably sample some edges of $H^*$ and add them to graph $H$. This will allow us to (approximately) preserve the cuts, while ensuring Properties (C3) and (C4).

We proceed in two steps. In the first step, we sample the edges in $E(H^*)$ to ensure that, for each $Q$-cluster $C \in Q$, for each group $U \in \mathcal{G}_C$, the number of remaining edges in $H^*$ whose tendrils terminate at edges in group $U$ is bounded by poly log log $n$, while the cut sizes only go down by a factor of $O(\log^4 n)$. This step requires a new constructive proof of Lovasz Local Lemma [54], which allows us to handle exponentially many bad events.

**Theorem 18** ([54]) Let $X$ be a finite set of mutually independent random variables in some probability space. Let $\mathcal{A}$ be a finite set of bad events determined by these variables. For each event $A \in \mathcal{A}$, let $\text{vbl}(A) \subseteq X$ be the unique minimal subset of variables determining $A$, and let $\Gamma(A) \subseteq \mathcal{A}$ be a subset of bad events $B$ such that $A \neq B$, but $\text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset$. Assume further that there is a constant $\epsilon \in (0,1)$ and an assignment of reals $x : \mathcal{A} \rightarrow (0, 1 - \epsilon)$ such
\[ \forall A \in \mathcal{A} : \Pr[A]^{1-\epsilon} \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)) \]  

Then there is an efficient randomized algorithm that w.h.p. finds an assignment to the variables of \( X \) such that none of the events in \( \mathcal{A} \) holds.

For each cluster \( C \in \mathcal{Q} \), for each group \( U \in \mathcal{G}_C \), we define the set \( S(C,U) \) of edges in \( E(H^*) \), consisting of all edges whose first tendril terminates at an edge of \( U \). Similarly, let \( S'(C,U) \) be the set of all edges whose second tendril terminates at an edge of \( U \). Recall that \( |U| \leq Z = O(\log^4 n) \) and that for each edge in \( U \), there are at most six tendrils terminating at it. So \( |S(C,U)| \leq 6Z \). We select each edge \( e \in E(H^*) \) with probability \( 1/Z \). For each \( C \in \mathcal{Q}, U \in \mathcal{G}_C \), we denote by \( \tilde{\beta}_{C,U} \) and \( \tilde{\beta}'_{C,U} \) the bad events that more than \( 100 \log Z \) edges in \( S(C,U) \) and \( S'(C,U) \) are selected, respectively. By Chernoff’s bound, the probability that \( \tilde{\beta}_{C,U} \) happens is at most \( 1/Z^{50} \) (and similarly for \( \tilde{\beta}'_{C,U} \)). For each subset \( W \subseteq V(H^*) \), we define a bad event \( \tilde{\beta}_W \) as the event that the number of selected edges in \( \delta_{H^*}(W) = E_{H^*}(W,V(H^*) \setminus W) \) is less than \( |\delta_{H^*}(W)|/2Z \). Notice that \( \Pr[\tilde{\beta}_W] \leq e^{-|\delta_{H^*}(W)|/10Z} \), which is at most \( n^{-|\delta_{H^*}(W)|/10Z \log n} \).

Now we are ready to define assignment \( x : \mathcal{A} \to (0, 1 - \epsilon) \) and argue that all these bad events do not happen. For events of the form \( \beta_{C,U} \) and \( \beta'_{C,U} \), define \( x(\beta_{C,U}) = x(\beta'_{C,U}) = 1/Z^{40} \). And for each event \( \tilde{\beta}_W \), let \( x(\tilde{\beta}_W) = e^{-|\delta_{H^*}(W)|/20Z} = n^{-|\delta_{H^*}(W)|/20Z \log n} \). We use the following theorem of Karger [62]

**Theorem 19 (Theorem 6.2 in [62])** For any half-integer \( k \), the number of cuts of weight at most \( kC \) in any graph is less than \( n^{2k} \), where \( C \) is the size of the minimum cut.

For convenience, we will distinguish between the dependencies \( \Gamma(A) \) of two different types. For each event \( A \in \mathcal{A} \), we define \( \Gamma'(A) \) to be the set of bad events \( B \) of the form \( \beta_{C,U} \) or \( \beta'_{C,U} \) such that \( vbl(A) \cap vbl(B) \neq \emptyset \). Similarly we define \( \Gamma''(A) \) to be the bad events \( \tilde{\beta}_W \) such that \( vbl(A) \cap vbl(\tilde{\beta}_W) \neq \emptyset \). Clearly, for any event \( A \in \mathcal{A} \), we have \( \Gamma(A) = \Gamma'(A) \cup \Gamma''(A) \). The following claim bounds the contribution from the second type of dependencies.
Claim 3  For any event $A$, 
\[
\prod_{\tilde{W} \in \Gamma''(A)} (1 - x(\tilde{W})) \geq \prod_{W \subseteq V(H^*)} (1 - x(\tilde{W})) \geq (1 - 1/n^2)
\]

Proof: Let $L_{\min}$ denote the size of the minimum cut in graph $H^*$. Then, 
\[
\prod_{W \subseteq V(H^*)} (1 - x(\tilde{W})) = \prod_{r \geq L_{\min}} \prod_{W:|\delta_H^+(W)|=r} (1 - x(\tilde{W})) \geq \prod_{r \geq L_{\min}} (1 - n^{-r/20Z \log n})^{n^4r/L_{\min}} \geq \prod_{r \geq L_{\min}} \left(1 - n^{4Z \log n + r/20Z \log n}/L_{\min}\right)
\]

Note that the second inequality follows from Theorem 19. If we choose $L = \Omega(\log^5 n)$ such that $L_{\min} \geq 160Z \log n$, we get that the product term is at least 
\[
\prod_{r \geq L_{\min}} (1 - n^{-r/40Z \log n}) \geq 1 - \sum_{r \geq L_{\min}} n^{-r/40Z \log n} \geq 1 - 1/n^2
\]

We now show that equation (2.1) holds for all bad events. Notice that each event $\beta_{C,U}$ depends on at most $6Z$ other events of the form $\beta_{C',U'}$, because the number of edges in $E(H^*)$ whose first tendrils terminate at edges of $U$ is at most $6Z$. Therefore, we have that 
\[
\prod_{A \in \Gamma'(\beta_{C,U})} (1 - x(A)) \geq (1 - 1/Z^{40})^{6Z} \geq (1 - 1/Z^{38}) \geq 1/2,
\]

and that 
\[
x(\beta_{C,U}) \prod_{A \in \Gamma(x(\beta_{C,U}))} (1 - x(A)) \geq \frac{1}{Z^{40}}(1 - n^{-2})(1/2) \geq \Pr[\beta_{C,U}]^{1/2}
\]

The analysis for events $\beta'_{C,U}$ is done similarly. Now we consider events of the form $\tilde{W}$. For each $W$, the event $\tilde{W}$ depends only on random variables on edges in $\delta_H^+(W)$, i.e. $|vbl(\tilde{W})| \leq |\delta_H^+(W)|$. Moreover, each edge $e \in \delta_H^+(W)$ has two tendril endpoints, so
there are at most $2|\delta^*_H(W)|$ events $\beta_{C,U}$ and $\beta'(C,U)$ that are dependent on $\tilde{\beta}_W$. In other words, $|\Gamma'(\tilde{\beta}_W)| \leq 2|\delta^*_H(W)|$, and therefore $\prod_{A \in \Gamma'(\tilde{\beta}_W)} (1 - x(A)) \geq (1 - 1/Z^{40})^2 |\delta^*_H(W)| \geq e^{-\frac{4|\delta^*_H(W)|}{Z^{40}}}$. Finally, we have

$$x(\tilde{\beta}_W) \prod_{A \in \Gamma(x(\tilde{\beta}_W))} (1 - x(A)) \geq e^{-\frac{|\delta^*_H(W)|}{20Z}} e^{-\frac{4|\delta^*_H(W)|}{Z^{40}}} (1 - \frac{1}{n^2}) \geq e^{-\frac{|\delta^*_H(W)|}{15Z}} \geq \Pr[\tilde{\beta}_W]^{0.99}.$$

This implies that the assignment $x : A \to (0,1)$ satisfies Equation (2.1) for the value $\epsilon = 0.01$, and therefore we have a polynomial-time randomized algorithm for finding the resulting graph $\tilde{H}$ with high probability. Our graph $\tilde{H}$ still satisfies properties (C1) and (C2) with parameter $\alpha^* = O(\log^4 n \log \log \log n)$, and for each cluster $C \in Q$, for each group $U \in G_C$, we have at most $O(\log Z) = O(\log \log n)$ edges in $E(\tilde{H})$ terminating at some edge of $U$.

In the second step, we further sample edges from graph $\tilde{H}$ to obtain our final graph $H$. We use the following theorem, which is a variant of Karger’s graph skeleton constructions [63, 64, 65].

**Theorem 20** Let $G = (V, E)$ be any graph, where the size of the minimum cut is $C$. Assume that the edges in $E$ are partitioned into disjoint subsets $E_1, E_2, \ldots, E_r$, of size at most $q$ each, where $q < C/(128 \log n)$. Construct a graph $G' = (V, E')$ as follows: for each group $E_i : 1 \leq i \leq r$, sample one edge $e \in E_i$ uniformly at random, and add it to $E'$. Then with probability at least $(1 - 1/n^3)$, for each partition $(A, B)$ of the set $V$ of vertices, $|E_{G'}(A, B)| \geq \frac{1}{2q} |E_G(A, B)|$.

**Proof:** We say that a cut $(A, B)$ is violated if $|E_{G'}(A, B)| < \frac{1}{2q} |E_G(A, B)|$. Consider some partition $(A, B)$ of the vertices of $V$, and denote $\hat{E} = E_G(A, B)$. For each $i : 1 \leq i \leq r$, let $\hat{E}_i = E_i \cap \hat{E}$. For each $1 \leq i \leq r$, we define an indicator random variable $y_i$, which is set to 1 iff an edge of $\hat{E}_i$ has been added to $E'$. Then $\Pr[y_i = 1] = |\hat{E}_i|/|E_i| \geq |\hat{E}_i|/q$, and
variables \( \{y_i\}_{i=1}^r \) are mutually independent. Then \( |E_{G'}(A, B)| = \sum_i y_i \), and the expectation \( \mu = \mathbb{E} \left[ \sum_i y_i \right] \geq |\hat{E}|/q \). Using the Chernoff bound,

\[
\Pr \left[ |E_{G'}(A, B)| < \frac{1}{2q} |E_G(A, B)| \right] = \Pr \left[ \sum_i y_i < \mu/2 \right] \leq e^{-\mu/8} \leq e^{-|\hat{E}|/(8q)}
\]

For each \( j : 1 \leq j \leq 2 \log n \), we let \( C_j \) denote the collection of all cuts \((A, B)\) with \( 2^{j-1}C < |E_G(A, B)| \leq 2^j C \). From Karger’s theorem, \( |C_j| < n^{2j+1} \). From the above calculations, the probability that a given cut \((A, B) \in C_j\) is violated, is at most \( e^{-2^{j-1}C/(8q)} = e^{-2^{j-4}C/q} \).

Using the union bound, the probability that any cut in \( C_j \) is violated is at most

\[
e^{-2^{j-4}C/q} \cdot n^{2j+1} = e^{-2^{j-4}C/q + 2^{j+1} \ln n} \leq e^{-2^{j-4}C/q} \leq e^{-2^{j+1} \ln n}
\]

if \( q < \frac{C}{2^r \log n} \). Using a union bound over all \( 1 \leq j \leq 2 \log n \), we get that the with probability at least \( (1 - 1/n^3) \), none of the cuts is violated.

We call the procedure outlined in the above theorem an edge sampling procedure. Once we specify the partition \( E_1, \ldots, E_r \) of the edges of our graph, the edge sampling procedure is fixed. We now perform four rounds of the edge sampling procedure, each of which uses a different partition of the edges of \( \tilde{H} \). The edges that survive all four rounds of sampling will then be added to graph \( H \).

**First partition** For each critical cluster \( C \in Q \), let \( G_C \) be the grouping of the edges of \( \text{out}(C) \). For each cluster \( C \in Q \), for each group \( U \in G_C \), we define a set \( S(C, U) \), consisting of all edges in \( E(\tilde{H}) \) whose first tendril terminates at an edge of \( U \). Notice that all sets \( S(C, U) \) are disjoint. The previous step guarantees that \( |S(C, U)| \leq O(\log \log n) \). Let \( \mathcal{E}_1 \)
denote the resulting partition of the edges $E(\tilde{H})$.

**Second partition** The second partition, $\mathcal{E}_2$, is defined exactly like the first partition, except that now we group by the second tendril, and not by the first tendril. Let $\mathcal{E}_2$ denote the resulting grouping.

In order to define the third and the fourth partitions, we consider the J-clusters $C \in \mathcal{J}$. Intuitively, the edges of the expander $X_C$ define demands on the edges of $\text{out}(C)$, that we would like to be able to route inside $C$ with small congestion. We cannot do it directly, and instead employ Corollary 9 for routing in small clusters. For each J-cluster $C \in \mathcal{J}$, let $\mathcal{G}_C$ be the partition of the edges of $\text{out}(C)$ given by Corollary 9. Recall that the size of each group is $z = \text{poly log log } n$, and any set of $(\gamma, \mathcal{G}_C)$-restricted integral demands on $\text{out}(C)$ can be routed integrally with constant congestion inside $C$.

**Third partition** The third partition, $\mathcal{E}_3$ is defined as follows. Every edge $e \in E^0$ is placed in a separate group. For each J-cluster $C \in \mathcal{J}$, we group the edges of $E^C$ as follows. For each set $U \in \mathcal{G}_C$, we create a group $S(C, U) \subseteq E^C$, that contains all edges $e' \in E^C$, for which $p_1(e') \in U$. This defines a partition of edges of $E^C$ into groups of size $\text{poly log log } n$. Each such group is then added to $\mathcal{E}_3$.

**Fourth partition** The fourth partition, $\mathcal{E}_4$, is defined exactly like the third partition, except that we are using $p_2(e')$ instead of $p_1(e')$ to group the edges of $E^C$.

We perform the edge sampling procedure four times, once for each partition $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4$. Edges that are selected by all four sampling procedures then become the edges of $H$. The sizes of groups in these partitions are $\text{poly log log } n$. From Theorem 20, w.h.p., for each partition $(A, B)$ of $V(H) = V(\tilde{H})$, $|E_H(A, B)| = \Omega\left(\frac{|E_{\tilde{H}}(A, B)|}{\text{poly log log } n}\right)$. Combining this with Theorem 17 and Corollary 3 immediately implies Properties (C1) and (C2). Our sampling using the partitions $\mathcal{E}_1$ and $\mathcal{E}_2$ immediately implies Property (C4). In order to establish Property (C3), consider any edge $e = (v_C, v_{C'})$ in graph $H$. If $e$ connects two critical
clusters, then we define $P_e = \{e\}$. Otherwise, assume that $e \in E^{C''}$ for some $J$-cluster $C''$. Let $\hat{E}$ be the subset of edges of $E^{C''}$ that belong to $H$. Then these edges define a set of $(2, G_{C''})$-restricted integral demands on the edges of $\text{out}_G(C'')$. From Corollary 9, we can route these demands inside $C''$ with constant congestion. For each edge $e \in \hat{E}$, let $P_{C''}(e)$ be the corresponding path in this routing.

Consider now some edge $e \in \hat{E}$. We define $P_e$ to be the concatenation of $\tau_1(e)$, $P_{C''}(e)$, and $\tau_2(e)$. Since all tendrils cause a constant congestion in $G$, and for each $J$-cluster $C''$, the routing of the demands on $\text{out}_G(C'')$ is performed with constant congestion, the final congestion of the set $\mathcal{P}_H^E = \{P_e \mid e \in E(H)\}$ of paths is bounded by a constant.
CHAPTER 3
BASIC INTEGRAL CONCURRENT FLOW

The goal of this section is to prove Theorem 1. We start by proving the theorem for the special case where all demands are uniform, and all edge capacities are unit. That is, we are given a subset \( \mathcal{M} \subseteq T \times T \) of the terminal pairs, and for each pair \( (t, t') \in \mathcal{M}, D(t, t') = D \), while all other demands are 0. We extend this algorithm to handle arbitrary demands and edge capacities in Section 5.1. We set the parameter \( L = \Theta(\log^{20} n \text{ polylog log } n) \), and we define its exact value later.

3.1 The Algorithm

Assume first that in the input instance \( G \), any cut separating the set \( T \) of terminals has value at least \( L \). We show in the next theorem, that in this case we can find an integral solution of value \( \lambda_{\text{OPT}}/\text{polylog } n \) with constant congestion to instance \((G, D)\). The main idea is to use Theorem 15 to construct a graph \( H \), where the initial set \( Q_0 \) of critical clusters is \( Q_0 = \{\{t\} \mid t \in T\} \). We then split \( H \) into \( \text{polylog } n \) sub-graphs using the procedure outlined in Section 2, and use the algorithm of Rao and Zhou [79] to route a polylogarithmic fraction of the demand pairs in each resulting subgraph. The proof of the next theorem appears later in Section 3.2.

**Theorem 21** Let \( G \) be an instance of basic-lCF with unit edge capacities and a set \( D \) of uniform demands over the set \( T \) of terminals. Assume that every cut separating the terminals in \( T \) has size at least \( L = \Theta(\log^{20} n \text{ polylog log } n) \). Then there is an efficient randomized algorithm that w.h.p. finds an integral solution of value \( \lambda_{\text{OPT}}/\beta \) with constant congestion, where \( \beta = O(\log^{26} n \text{ polylog log } n) \) and \( \lambda_{\text{OPT}} \) is the value of the optimal fractional solution.

In general, graph \( G \) may contain small cuts that separate its terminals. We get around this problem as follows. For each subset \( S \subseteq V \) of vertices, let \( T_S = S \cap T \) be the subset
of terminals contained in $S$. We say that $S$ is a **good subset** iff (1) Any cut in graph $G[S]$ separating the terminals in $\mathcal{T}_S$ has value at least $L$; and (2) $|\text{out}(S)| \leq L \log k$. We first show that we can efficiently compute a good set $S$ of vertices in graph $G$. We then decompose the set $\mathcal{D}$ of demands into two subsets: $\mathcal{D}_S$ containing the demands for all pairs contained in $\mathcal{T}_S$, and $\mathcal{D}'$ containing the demands for all other pairs. Next, we apply Theorem 21 to instance $(G[S], \mathcal{D}_S)$, obtaining a collection $\mathcal{P}'$ of paths, and solve the problem recursively on instance $(G, \mathcal{D}')$, obtaining a collection $\mathcal{P}''$ of paths. Our final step is to carefully combine the two sets of paths to obtain the final solution $\mathcal{P}$. We start with the following lemma that allows us to find a good subset $S$ of vertices efficiently.

**Lemma 6** Let $(G, \mathcal{D})$ be a basic-ICF instance with uniform demands and unit edge capacities, and a set $\mathcal{T}$ of terminals, where $|\mathcal{T}| \leq k$. Then there is an efficient algorithm that either finds a good set $S \subseteq V(G)$ of vertices, or establishes that every cut $(A, B)$ separating the terminals of $\mathcal{T}$ in $G$ has value $|E_G(A, B)| \geq L$.

**Proof:** We start with $S = V(G)$, and then perform a number of iterations. Let $G' = G[S]$, and let $\mathcal{T}_S = \mathcal{T} \cap S$. Let $(A, B)$ be the minimum cut separating the terminals in $\mathcal{T}_S$ in graph $G'$, and assume w.l.o.g. that $|A \cap \mathcal{T}_S| \leq |B \cap \mathcal{T}_S|$. If $|E_{G'}(A, B)| < L$, then we set $S = A$, and continue to the next iteration. Otherwise, we output $S$ as a good set. (If $S = V(G)$, then we declare that every cut separating the terminals in $\mathcal{T}$ has value at least $L$.)

Clearly, if $S$ is the final set that the algorithm outputs, then every cut in graph $G[S]$ separating the set $\mathcal{T} \cap S$ of terminals has value at least $L$. It only remains to show that $|\text{out}_G(S)| \leq L \log k$. Since $|\mathcal{T}| \leq k$, and the number of terminals contained in set $S$ goes down by a factor of at least 2 in every iteration, there are at most $\log k$ iterations. In each iteration, at most $L$ edges are deleted. Therefore, $|\text{out}_G(S)| \leq L \log k$.

We use the following theorem, whose proof appears below in Section 3.3, to combine the solutions to the two sub-instances. In this theorem, we assume that we are given a good vertex set $S$, $\mathcal{T}_S = \mathcal{T} \cap S$, and $\mathcal{M}_S \subseteq \mathcal{M}$ is the subset of the demand pairs contained in $S$. 
We assume w.l.o.g. that $M_S = \{(s_1, t_1), \ldots, (s_{k'}, t_{k'})\}$.

**Theorem 22** Suppose we are given a good vertex set $S$, and $M_S \subseteq M$ as above. Assume further that for each $1 \leq i \leq k'$, we are given a set $P_i$ of $N$ paths connecting $s_i$ to $t_i$, such that all paths in set $P' = \bigcup_{i=1}^{k'} P_i$ are contained in $G[S]$, and set $P'$ causes congestion at most $\gamma$ in $G[S]$. Let $P''$ be any set of paths in graph $G$, where each path in $P''$ connects some pair $(s, t) \in M \setminus M_S$, and the congestion caused by the paths in $P''$ is at most $\gamma$. Then we can efficiently find, for each $1 \leq i \leq k'$, a subset $P_i^* \subseteq P_i$ of at least $N - 2L\gamma \log n$ paths, and for each $P \in P''$, find a path $\hat{P}$ connecting the same pair of vertices as $P$, such that the total congestion caused by the set $\left( \bigcup_{i=1}^{k'} P_i^* \right) \cup \{ \hat{P} \mid P \in P'' \}$ of paths is at most $\gamma$ in graph $G$.

We denote this algorithm by REROUTE($P', P''$), and its output is denoted by $\hat{P}' = \bigcup_{i=1}^{k'} P_i^*$, and $\hat{P}'' = \{ \hat{P} \mid P \in P'' \}$. We now complete the description of our algorithm, that we call RECURSIVROUTING.

We assume that we are given a graph $G$, a set $T$ of at most $k$ terminals, a set $M$ of demand pairs, and a real number $D > 0$, such that for each pair $(t, t') \in M$, we can send $\lambda_{opt} \cdot D$ flow units simultaneously in $G$ with no congestion. If no cut of size less than $L$ separates the terminals of $T$, then we use Theorem 21 to find a set $P$ of paths and return $P$. Otherwise, we find a good vertex set $S \subseteq V(G)$ using Lemma 6. Let $T_S = T \cap S$, $M_S \subseteq M$ the subset of pairs contained in $T_S$, $M' = M \setminus M_S$. We then apply Theorem 21 to $(G[S], M_S, D)$ to obtain a collection $P'$ of paths, and invoke RECURSIVROUTING on $(G, M', D)$ to obtain a set $P''$ of paths. Finally, we apply procedure REROUTE to sets $P', P''$ of paths to obtain the collections $\hat{P}', \hat{P}''$ of paths, and return $P = \hat{P}' \cup \hat{P}''$.

Let $\beta$ be the parameter from Theorem 21, and let $\gamma$ be the congestion it guarantees. We can assume w.l.o.g. that $\frac{\lambda_{opt}}{\beta} D \geq 4L\gamma \log n$, since otherwise $\lfloor \frac{\lambda_{opt}}{4L\gamma \log n} D \rfloor = 0$, and $P = \emptyset$ is a (poly log)-approximate solution to the problem. We now prove that procedure RECURSIVROUTING produces a solution of value $\frac{\lambda_{opt}}{4L\gamma}$ and congestion at most $\gamma$. 

62
Lemma 7 Let $\mathcal{P}$ be the output of procedure RecursiveRouting. Then for every pair $(s, t) \in \mathcal{M}$, at least $\lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil$ paths connect $s$ to $t$ in $\mathcal{P}$, and the paths in $\mathcal{P}$ cause congestion at most $\gamma$ in $G$.

Proof: The proof is by induction on the recursion depth. In the base case, when no cut of size less than $L$ separates the terminals of $T$ in $G$, the correctness follows directly from Theorem 21.

Otherwise, consider the set $\mathcal{P}'$ of paths. For each pair $(s, t) \in \mathcal{M}$, let $\mathcal{P}'(s, t) \subseteq \mathcal{P}'$ be the subset of paths connecting $s$ to $t$ in $\mathcal{P}'$. From Theorem 21, we are guaranteed that $|\mathcal{P}'(s, t)| \geq \lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil$ for each $(s, t) \in \mathcal{M}$, and the paths in $\mathcal{P}'$ cause congestion at most $\gamma$ in $G[S]$.

Consider now the set $\mathcal{P}''$ of paths. For each pair $(s, t) \in \mathcal{M}'$, let $\mathcal{P}''(s, t) \subseteq \mathcal{P}''$ be the subset of paths connecting $s$ to $t$ in $\mathcal{P}''$. From the induction hypothesis, $|\mathcal{P}''(s, t)| \geq \lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil$ for all $(s, t) \in \mathcal{M}'$, and the paths in $\mathcal{P}''$ cause congestion at most $\gamma$ in $G$.

Consider now the final set $\mathcal{P} = \tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$ of paths returned by the algorithm. From Theorem 22, the paths in $\mathcal{P}$ cause congestion at most $\gamma$, as required. For each pair $(s, t) \in \mathcal{M}$, the set $\tilde{\mathcal{P}}'$ of paths contains at least $\lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil - 2L\gamma \log n \geq \lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil$ paths. For each pair $(s, t) \in \mathcal{M}$, if $\tilde{\mathcal{P}}''(s, t)$ is the subset of paths of $\tilde{\mathcal{P}}''$ connecting $s$ to $t$, then $|\tilde{\mathcal{P}}''(s, t)| = |\mathcal{P}''(s, t)| \geq \lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil$, since each path in $\mathcal{P}''$ is replaced by a path connecting the same pair of vertices in $\tilde{\mathcal{P}}''$. Therefore, each pair $(s, t) \in \mathcal{M}$ is connected by at least $\lceil \frac{\lambda_{\text{opt}}}{4\beta} D \rceil$ paths in $\mathcal{P}$.

This completes the proof of Theorem 1 for uniform demands and unit edge capacities, except for the proofs of Theorems 21 and 22 that appear below. In Section 5.1 we extend this algorithm to arbitrary edge capacities and demands using standard techniques.
3.2 Proof of Theorem 21

We assume that we are given a collection \( \mathcal{M} \subseteq \mathcal{T} \times \mathcal{T} \) of terminal pairs and a value \( D > 0 \), such that for each pair \((t, t') \in \mathcal{M}\), \( D(t, t') = 1 \), and all other demands are 0.

We start by applying Theorem 15 to graph \( G \), where we use the threshold \( L \) from the statement of Theorem 21 for the definition of small clusters, and the initial set of critical clusters is \( Q_0 = \{ \{t\} \mid t \in \mathcal{T} \} \). It is easy to see that each cluster in \( Q_0 \) is a critical cluster, and any cut separating the clusters in \( Q_0 \) in graph \( G \) is large. Let \( H \) be the resulting graph guaranteed by Theorem 15.

Since every terminal in \( \mathcal{T} \) is mapped to a separate vertex of \( H \), we can view \( D \) as a set of demands for graph \( H \). We now focus on finding a solution to the ICF problem instance in graph \( H \), and later transform it into a solution in the original graph \( G \). We use the following theorem, due to Rao and Zhou [79].

**Theorem 23 ([79])** Let \( G' \) be any \( n \)-vertex graph, and let \( \mathcal{M}' = \{(s_1, t_1), \ldots, (s_k, t_k)\} \) be any set of source-sink pairs in \( G' \). Assume further that the value of the global minimum cut in graph \( G' \) is at least \( L_{RZ} = \Omega(\log^5 n) \), and there is a fractional multi-commodity flow, where for each \( 1 \leq i \leq k \), source \( s_i \) sends \( f_i \leq 1 \) flow units to sink \( t_i \), \( \sum_{i=1}^{k} f_i = F \), and the flow causes no congestion in \( G' \). Then there is an efficient randomized algorithm that w.h.p. finds a collection \( \mathcal{M}^* \subseteq \mathcal{M}' \) of at least \( F/\alpha_{RZ} \) demand pairs, and for each pair \((s, t) \in \mathcal{M}^* \), a path \( P(s, t) \) connecting \( s \) to \( t \) in \( G' \), such that the paths in the set \( \mathcal{P}^* = \{P(s, t) \mid (s, t) \in \mathcal{M}^*\} \) are edge-disjoint, and \( \alpha_{RZ} = O(\log^{10} n) \).

Let \( x = 8\alpha_{RZ} \cdot \log n = O(\log^{11} n) \). We set \( L = 2\alpha^* \cdot x \cdot L_{RZ} = O(\log^{20} n \polylog \log n) \).

We split graph \( H \) into \( x \) graphs \( H_1, \ldots, H_x \), as follows. For each \( 1 \leq i \leq x \), we set \( V(H_i) = V(H) \). In order to define the edge sets of graphs \( H_i \), each edge \( e \in E \), chooses an index \( 1 \leq i \leq x \) independently uniformly at random, and is then added to \( E(H_i) \). This completes the definition of the graphs \( H_i \). Given any partition \((A, B)\) of the vertices of \( V(H) \), let \( \text{cut}_G(A, B) \) denote the value of the minimum cut \(|E_G(A', B')|\) in graph \( G \), such that for
each \( v_Q \in A, Q \subseteq A' \), and for each \( v_Q \in B, Q \subseteq B' \). Recall that Theorem 15 guarantees that the size of the minimum cut in \( H \) is at least \( L/\alpha^* \), and for each partition \((A,B)\) of \( V(H) \), \( \text{cut}_G(A,B) \leq \alpha^* \cdot |E_H(A,B)| \). From Theorem 16, w.h.p., for each \( 1 \leq i \leq x \), we have that:

- The value of the minimum cut in \( H_i \) is at least \( L/2 \alpha^*x = L_{RZ} \).
- For any cut \((A,B)\) of \( V(H_i) \), \( |E_H(A,B)| \geq \frac{\text{cut}_G(A,B)}{2x\alpha^*} \).

From now on we assume that both properties hold for each graph \( H_i \). We then obtain the following observation, whose proof appears in the full version of the paper.

**Observation 2** For each \( 1 \leq i \leq x \), there is a fractional solution to the instance \((H_i, D)\) of basic-ICF of value \( \frac{\lambda_{\text{opt}}}{2x\alpha^* \beta_{\text{FCG}}} \) and no congestion.

**Proof:** Assume otherwise. Then the value of the maximum concurrent flow in graph \( H_i \) for the set \( D \) of demands is less than \( \frac{\lambda_{\text{opt}}}{2x\alpha^* \beta_{\text{FCG}}} \).

Recall that in the non-uniform sparsest cut problem, we are given a graph \( G \) and a set \( D \) of demands. The goal is to find a cut \((A,B)\) in \( G \) of minimum sparsity, where the sparsity of the cut \((A,B)\) is \( \frac{|E(A,B)|}{D_G(A,B)} \). Here, \( D_G(A,B) \) is the sum of demands \( D(s,t) \) for all pairs \((s,t)\) with \( s \in A, t \in B \). This problem is the dual of the maximum concurrent flow problem in graph \( G \) with the set \( D \) of demands. In particular, if the value of the maximum concurrent flow is at most \( \lambda \), then the value of the minimum sparsest cut is at most \( \lambda \cdot \beta_{\text{FCG}} \).

We set up an instance of the non-uniform sparsest cut problem on graph \( H_i \) with the set \( D \) of demands. Then there is a cut \((A,B)\) in \( H_i \), with \( \frac{|E_{H_i}(A,B)|}{D_{H_i}(A,B)} < \frac{\lambda_{\text{opt}}}{2x\alpha^*} \). Let \((A',B')\) be the minimum cut in graph \( G \), where for each \( v_Q \in A, Q \subseteq A' \), and for each \( v_Q \in B, Q \subseteq B' \). Then \( |E_{G}(A',B')| = \text{cut}_G(A,B) \leq 2x\alpha^*|E_{H_i}(A,B)| \), while \( D_{G}(A',B') = D_{H_i}(A,B) \). Therefore,

\[
\frac{|E_{G}(A',B')|}{D_{G}(A',B')} \leq 2x\alpha^* \frac{|E_{H_i}(A,B)|}{D_{H_i}(A,B)} < \lambda_{\text{opt}}.
\]
This is impossible, since we have assumed that the value of the optimal fractional solution to the ICF instance \((G, D)\) is \(\lambda_{\text{OPT}}\).

In the rest of the algorithm, we apply the algorithm of Rao-Zhou to each of the graphs \(H_1, \ldots, H_x\) in turn. For each \(1 \leq i \leq x\), in the \(i\)th iteration we define a subset \(\mathcal{M}_i \subseteq \mathcal{M}\) of pairs of terminals (that are not satisfied yet), and we define the set \(\mathcal{D}_i\) of demands to be \(D_i(t, t') = D\) if \((t, t') \in \mathcal{M}_i\), and \(D_i(t, t') = 0\) otherwise. For the first iteration, \(\mathcal{M}_1 = \mathcal{M}\). We now describe an execution of iteration \(i \geq 1\).

Suppose we are given a set \(\mathcal{M}_i\) of terminal pairs and a corresponding set \(\mathcal{D}_i\) of demands. We construct a new collection \(\mathcal{M}'_i\) of source-sink pairs, where the demand for each pair is 1, as follows. For each pair \((t, t') \in \mathcal{M}_i\), we add \(N = \lfloor \frac{\lambda_{\text{OPT}}}{2\alpha^{*} \beta_{\text{FCG}}} \cdot D \rfloor\) copies of the pair \((t, t')\) to \(\mathcal{M}'_i\). We then apply Theorem 23 to the resulting graph and the set \(\mathcal{M}'_i\) of demand pairs. From Observation 2, there is a flow of value at least \(F_i = N \cdot |\mathcal{M}_i| = |\mathcal{M}'_i|\) in the resulting graph. Therefore, from Theorem 23, w.h.p. we obtain a collection \(\mathcal{P}_i\) of paths connecting the demand pairs in \(\mathcal{M}_i\) with no congestion, and \(|\mathcal{P}_i| \geq \frac{F_i}{\alpha_{\text{RZ}}} \geq \frac{N}{\alpha_{\text{RZ}}} |\mathcal{M}_i|\). We say that a pair \((t, t') \in \mathcal{M}_i\) of terminals is satisfied in iteration \(i\), iff the number of paths in \(\mathcal{P}_i\) connecting \(t\) to \(t'\) is at least \(\frac{N}{2\alpha_{\text{RZ}}}\). We then let \(\mathcal{M}_{i+1} \subseteq \mathcal{M}_i\) be the subset of terminal pairs that are not satisfied in iteration \(i\). This concludes the description of our algorithm for routing on graph \(H\). The key in its analysis is the following simple claim.

**Claim 4** For each \(1 \leq i \leq x\), \(|\mathcal{M}_{i+1}| \leq \left(1 - \frac{1}{2\alpha_{\text{RZ}}}\right) |\mathcal{M}_i|\).

**Proof:** Let \(\mathcal{M}_i^s \subseteq \mathcal{M}_i\) be the subset of demand pairs that are satisfied in iteration \(i\). It is enough to prove that \(|\mathcal{M}_i^s| \geq \frac{1}{2\alpha_{\text{RZ}}} |\mathcal{M}_i|\). Assume otherwise. A pair \((t, t') \in \mathcal{M}_i^s\) contributes at most \(N\) paths to \(\mathcal{P}_i\), while a pair \((t, t') \in \mathcal{M}_i \setminus \mathcal{M}_i^s\) contributes less than \(\frac{N}{2\alpha_{\text{RZ}}}\) paths. Therefore, if \(|\mathcal{M}_i^s| < \frac{1}{2\alpha_{\text{RZ}}} |\mathcal{M}_i|\), then:

\[
|\mathcal{P}_i| < |\mathcal{M}_i^s| \cdot N + |\mathcal{M}_i \setminus \mathcal{M}_i^s| \cdot \frac{N}{2\alpha_{\text{RZ}}} < \frac{N}{\alpha_{\text{RZ}}} |\mathcal{M}_i|,
\]

a contradiction. \(\square\)
Therefore, after $x = 8\alpha_{RZ} \cdot \log n$ iterations, we will obtain $\mathcal{M}_{x+1} = \emptyset$, and all demand pairs are satisfied. Recall that a demand pair is satisfied iff there are at least $\frac{N}{2\alpha_{RZ}} = \Omega\left(\frac{\lambda_{\text{OPT}}}{\alpha_{RZ}x^{0.5} \beta_{\text{FCG}}} \cdot D\right) = \Omega\left(\frac{\lambda_{\text{OPT}}}{\log^{20} n \text{poly} \log \log n} \cdot D\right)$ paths connecting them. Therefore, we have shown an integral solution to the ICF instance $(H, \mathcal{D})$ of value $\Omega\left(\frac{\lambda_{\text{OPT}}}{\log^{20} n \text{poly} \log \log n}\right)$ and no congestion.

We now show how to obtain an integral solution to the ICF instance $(G, \mathcal{D})$, of the same value and constant congestion. Let $\mathcal{P}^*$ be the set of paths in graph $H$ that we have obtained. We transform each path $P \in \mathcal{P}^*$ into a path $P'$ connecting the same pair of terminals in graph $G$. Recall that all terminals in $\mathcal{T}$ are vertices in both $G$ and $H$. For each edge $e = (v_Q, v_{Q'})$ on path $P$, we replace $e$ with the path $P_e$, connecting some vertex $u \in Q$ to some vertex $u' \in Q'$, guaranteed by Property (C3) of graph $H$. Once we process all edges on all paths $P \in \mathcal{P}^*$, we obtain, for each cluster $Q \in \mathcal{Q}$, a set $\mathcal{D}_Q$ of demands on the edges of out$(Q)$, that need to be routed inside the cluster $Q$. From Property (C4), this set of demands must be $(2, \mathcal{G}_Q)$-restricted. Combining Observation 1 with Theorem 10, we obtain an efficient randomized algorithm that w.h.p. routes the set $\mathcal{D}_Q$ of demands integrally inside $Q$ with constant congestion. For each path $P \in \mathcal{P}^*$, we can now combine the paths $P_e$ for $e \in P$ with the resulting routing inside the clusters $Q$ for each $v_Q \in P$ to obtain a path $P'$ in graph $G$ connecting the same pair of terminals as $P$. Since the set $\{P_e \mid e \in E(H)\}$ of paths causes a constant congestion in graph $G$ from Property (C3), the resulting set of paths causes a constant congestion in $G$.

### 3.3 Proof of Theorem 22

We use the following lemma as a subroutine.

**Lemma 8** Let $G'$ be any graph, and let $\mathcal{S}_1, \mathcal{S}_2$ be two sets of paths in $G'$, where the paths in each set are edge-disjoint (but the paths in $\mathcal{S}_1 \cup \mathcal{S}_2$ may share edges). Assume further that all paths in $\mathcal{S}_1$ originate at the same vertex $s$. Then we can efficiently find a subset $\mathcal{S}_1' \subseteq \mathcal{S}_1$
of at least \(|S_1| - 2|S_2|\) paths, and for each path \(P \in S_2\), another path \(\tilde{P}\) connecting the same pair of vertices as \(P\), such that, if we denote \(S_2' = \{\tilde{P} \mid P \in S_2\}\), then:

1. All paths in \(S_1' \cup S_2'\) are edge-disjoint.

2. Let \(E'\) and \(\tilde{E}\) be the sets of edges used by at least one path in \(S_1 \cup S_2\) and \(S_1' \cup S_2'\) respectively. Then \(\tilde{E} \subseteq E'\).

In other words, the lemma re-routes the paths in \(S_2\), using the paths in \(S_1\), and then chooses \(S_1'\) to be the subset of paths in \(S_1\) that do not share edges with the new re-routed paths in \(S_2'\). The rerouting guarantees that re-routed paths only overlap with at most \(2|S_2|\) paths in \(S_1\).

**Proof:** [Of Lemma 8]

The proof is very similar to arguments used by Conforti et al. [42]. Given any pair \((P, P')\) of paths, we say that paths \(P\) and \(P'\) intersect at edge \(e\), if both paths contain edge \(e\), and we say that \(P\) and \(P'\) intersect iff they share any edge.

We set up an instance of the stable matching problem in a multi-graph. In this problem, we are given a complete bipartite multigraph \(G = (A, B, E)\), where \(|A| = |B|\). Each vertex \(v \in A \cup B\) specifies an ordering \(R_v\) of the edges adjacent to \(v\) in \(G\). A complete matching \(M\) between the vertices of \(A\) and \(B\) is called stable iff, for every edge \(e = (a, b) \in E \setminus M\), the following holds. Let \(e_a, e_b\) be the edges adjacent to \(a\) and \(b\) respectively in \(M\). Then either \(a\) prefers \(e_a\) over \(e\), or \(b\) prefers \(e_b\) over \(e\). Conforti et al. [42], generalizing the famous theorem of Gale and Shapley [52], show an efficient algorithm to find a perfect stable matching \(M\) in any such multigraph.

Given the sets \(S_1, S_2\) of paths, we set up an instance of the stable matching problem as follows. Set \(A\) contains a vertex \(a(P)\) for each path \(P \in S_1\). For each path \(P \in S_2\), if \(x, y\) are the two endpoints of \(P\), then we add two vertices \(b(P, x)\) and \(b(P, y)\) to \(B\). In order to ensure that \(|A| = |B|\), we add dummy vertices to \(B\) as needed.
For each pair $P \in S_1$, $P' \in S_2$ of paths, for each edge $e$ that these paths share, we add two edges $(a(P), b(P', x))$ and $(a(P), b(P', y))$, where $x$ and $y$ are the endpoints of $P'$, and we think of these new edges as representing the edge $e$. We add additional dummy edges as needed to turn the graph into a complete bipartite graph.

Finally, we define preference lists for vertices in $A$ and $B$. For each vertex $a(P) \in A$, the edges incident to $a(P)$ are ordered according to the order in which they appear on path $P$, starting from $s$. The dummy edges incident to $a(P)$ are ordered arbitrarily at the end of the list.

Consider now some vertex $b(P, x) \in B$. We again order the edges incident to $b(P, x)$ according to the order in which their corresponding edges appear on the path $P$, when we traverse $P$ starting from $x$. The dummy edges incident on $b(P, x)$ are added at the end of the list in an arbitrary order. The preference list of the vertex $b(P, y)$ is defined similarly, except that now we traverse $P$ starting from $y$. Finally, the preference lists of the dummy vertices are arbitrary.

Let $M$ be any perfect stable matching in the resulting graph. We let $S'_1 \subseteq S_1$ be the subset of paths that are matched to the dummy vertices. Clearly, $|S'_1| \geq |S_1| - 2|S_2|$. For each path $P \in S_2$, we now define a path $\tilde{P}$, as follows. Let $x, y$ be the two endpoints of $P$. If at least one of the vertices $b(P, x), b(P, y)$ participates in $M$ via a dummy edge, then we let $\tilde{P} = P$, and we say that $P$ is of type 1. Otherwise, let $e, e'$ be the edges of $M$ incident on $b(P, x)$ and $b(P, y)$, respectively, and let $P_1, P_2 \in S_1$ be two paths such that $a(P_1)$ is the other endpoint of $e$ and $a(P_2)$ is the other endpoint of $e'$. Let $e_1, e_2$ be the edges of the original graph that the edges $e, e'$ represent. Let $\sigma_1(P)$ be the segment of $P$ from $x$ to $e$; $\sigma_2(P)$ the segment of $P_1$ from $e$ to $s$; $\sigma_3(P)$ the segment of $P_2$ from $s$ to $e'$; and $\sigma_4(P)$ the segment of $P$ from $e'$ to $y$. We set $\tilde{P}$ be the concatenation of $\sigma_1(P), \sigma_2(P), \sigma_3(P), \sigma_4(P)$, and we say that $P$ is of type 2. Let $S'_2 = \{ \tilde{P} | P \in S_2 \}$. It now only remains to show that all paths in $S'_1 \cup S'_2$ are edge-disjoint. It is immediate that the paths in $S'_1$ are edge-disjoint, since the paths in $S_1$ were edge-disjoint. We complete the proof in the following two claims.
Claim 5 All paths in $S'_2$ are edge-disjoint.

Proof: Assume otherwise, and let $\tilde{P}, \tilde{P}' \in S'_2$ be any pair of paths that share an edge, say $e$. First, it is impossible that both $P$ and $P'$ are of type 1, since then $P, P' \in S_2$, and so they must be edge-disjoint. So at least one of the two paths must be of type 2. Assume w.l.o.g. that it is $P$, and consider the four segments $\sigma_1(P), \sigma_2(P), \sigma_3(P), \sigma_4(P)$ of $\tilde{P}$, and the two edges $e_1, e_2$ that we have defined above. Let $P_1$ and $P_2$ be the paths in $S_1$ on which the segments $\sigma_2(P), \sigma_3(P)$ lie.

If $P'$ is of type 1, then it can only intersect $\sigma_2(P)$ or $\sigma_3(P)$, as the paths $P$ and $P'$ are edge-disjoint. Assume w.l.o.g. that $P'$ intersects $\sigma_2(P)$, and let $e'$ be any edge that they share. Let $x'$ be the endpoint of $P'$ such that the edge incident to $b(P', x')$ in $M$ is a dummy edge. Then $b(P', x')$ prefers $e'$ to its current matching, and $a(P_1)$ prefers $e'$ to its current matching as well, as $e'$ lies before $e_1$ on path $P'$, a contradiction.

Assume now that $P'$ is of type 2, and consider the segments $\sigma_1(P'), \sigma_2(P'), \sigma_3(P'), \sigma_4(P')$ of $\tilde{P}'$. Since the sets $S_1, S_2$ of paths are edge-disjoint, the only possibilities are that either one of the segments $\sigma_1(P), \sigma_4(P)$ intersects one of the segments $\sigma_2(P'), \sigma_3(P')$, or one of the segments $\sigma_1(P'), \sigma_4(P')$ intersects one of the segments $\sigma_2(P), \sigma_3(P)$. Assume w.l.o.g. that $\sigma_1(P)$ shares an edge $e$ with $\sigma_2(P')$. Let $x$ be the endpoint of $P$ to which $\sigma_1(P)$ is adjacent, and let $e_1$ be the last edge on $\sigma_1(P)$, and let $P_1 \in S_1$ be the path that shares $e_1$ with $P$. Then vertex $b(P, x)$ prefers the edge $e$ to its current matching, as it appears before $e_1$ on $P$, starting from $x$. Similarly, $a(P_1)$ prefers $e$ to its current matching, a contradiction.

\[ \Box \]

Claim 6 Paths in $S'_1$ and $S'_2$ are edge-disjoint from each other.

Proof: Assume otherwise, and let $P \in S'_1, P' \in S'_2$ be two paths that share an edge $e$. It is impossible that $P'$ is of type 1: otherwise, for some endpoint $x$ of $P'$, $b(P', x)$ is adjacent to a dummy edge in $M$, and $a(P)$ is also adjacent to a dummy edge in $M$, while both of them prefer $e$, a contradiction. Therefore, $P'$ is of type 2. Consider the four segments
\(\sigma_1(P'), \sigma_2(P'), \sigma_3(P'), \sigma_4(P')\). Since the paths in each set \(S_1\) and \(S_2\) are edge-disjoint, the only possibility is that \(e\) belongs to either \(\sigma_1(P')\), or to \(\sigma_4(P')\). Assume w.l.o.g. that \(e \in \sigma_1(P')\). Let \(x\) be the endpoint of \(P'\) to which \(\sigma_1(P')\) is adjacent. Then \(b(P', x)\) prefers \(e\) to its current matching, and similarly \(a(P)\) prefers \(e\) to its current matching, a contradiction. 

We are now ready to complete the proof of Theorem 22. We build a graph \(G'\) from graph \(G\), by replacing each edge of \(G\) with \(\gamma\) parallel edges. It is enough to define the subsets \(P_i^* \subseteq P_i\), and the paths \(\bar{P}\) for \(P \in P''\) in graph \(G'\), such that, in the resulting set \(\bar{P}' \cup \bar{P}'',\) all paths are edge-disjoint. From now on we focus on finding these path sets in graph \(G'\).

We perform \(k'\) iterations, where in iteration \(i\) we process the paths in set \(P_i\), and define a subset \(P_i^* \subseteq P_i\). In each iteration, we may also change the paths in \(P''\), by replacing some of these paths with paths that have the same endpoints as the original paths (we call this process re-routing). We maintain the invariant that at the beginning of iteration \(i\), the paths in set \(P'' \cup \left(\bigcup_{i=1}^{i-1} P_i^*\right)\) are edge-disjoint in \(G'\). We now describe the \(i\)th iteration, for \(1 \leq i \leq k'\).

Let \(S_1 = P_i\), and let \(S_2\) be the collection of consecutive segments of paths in \(P''\) that are contained in \(S\). The sets \(S_1\) and \(S_2\) of paths are both edge-disjoint in graph \(G'\), and so we can apply Lemma 8 to them, obtaining the sets \(S_1'\) and \(S_2'\) of paths. We then set \(P_i^* = S_1'\), and modify every path \(P'' \in P''\) by removing the segments of \(S_2\) from it, and adding the corresponding segments of \(S_2'\) instead. Let \(P''\) be the collection of the resulting paths. Clearly, the paths in \(P''\) connect the same pairs of terminals as the original paths, and they continue to be edge-disjoint in \(G'\) (since the re-routing was only performed inside the graph \(G'[S]\)). Moreover, since the paths in all sets \(P_1, \ldots, P_i\) are edge-disjoint, and we have only used the edges of the paths in \(P_i\) to re-route the paths in \(P''\), the paths in set \(P'' \cup \left(\bigcup_{i=1}^{i-1} P_i^*\right)\) are edge-disjoint in \(G'\). Finally, observe that \(|S_2| \leq \gamma L \log n\), since \(|\text{out}_G(S)| \leq L \log n\), and every path in \(S_2\) contains at least one edge of \(\text{out}_{G'}(S)\). Therefore, \(|P_i^*| \geq |P_i| - 2\gamma \log n\). Once we process all sets \(P_1, \ldots, P_{k'}\), our output is \(\{P_i^*\}_{i=1}^{k'}\), and we
output the final set $\mathcal{P}''$ that contains a re-routed path $\tilde{P}$ for each path $P$ in the original set.

### 3.4 Hardness of basic-ICF

In this section we prove Theorem 2, by performing a simple reduction from the Congestion Minimization problem. We use the following theorem, due to Andrews and Zhang [8].

**Theorem 24** Let $G$ be any $n$-vertex graph with unit edge capacities, and $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ be a collection of source-sink pairs. Then, unless $\text{NP} \subseteq \text{ZPTIME}(n^{\text{poly log } n})$, no efficient algorithm can distinguish between the following two cases:

- **(Yes-Instance):** There is a collection $\mathcal{P}$ of paths that causes congestion 1, and for each $1 \leq i \leq k$, there is a path connecting $s_i$ to $t_i$ in $\mathcal{P}$.

- **(No-Instance):** For any collection $\mathcal{P}$ of paths that contains, for each $1 \leq i \leq k$, a path connecting $s_i$ to $t_i$, the congestion due to $\mathcal{P}$ is at least $\Omega(\log \log n / \log \log \log n)$.

Let $G$ be any $n$-vertex graph with unit edge capacities and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs. We build a new graph $G'$, where we replace every edge in $G$ by $D$ parallel edges. Observe that if $G$ is a Yes-Instance, then there is a collection $\mathcal{P}$ of paths that causes congestion 1 in $G$, and every demand pair $(s_i, t_i) \in \mathcal{M}$ is connected by a path in $\mathcal{P}$. We then immediately obtain a collection $\mathcal{P}'$ of paths in graph $G'$ that causes congestion 1, and every demand pair $(s_i, t_i)$ is connected by $D$ paths, by simply taking $D$ copies of each path in $\mathcal{P}$. Assume now that $G$ is a No-Instance, and assume for contradiction that there is a collection $\mathcal{P}'$ of paths in graph $G'$ that causes congestion at most $c$, and every demand pair $(s_i, t_i)$ is connected by at least one path in $\mathcal{P}'$. Then set $\mathcal{P}'$ of paths defines a collection $\mathcal{P}$ of paths in graph $G$, that causes congestion at most $Dc$, and every demand pair $(s_i, t_i)$ is connected by at least one path in $\mathcal{P}$.

From Theorem 24, no efficient algorithm can distinguish between the case where there is a collection $\mathcal{P}'$ of edge-disjoint paths in $G'$, where every demand pair is connected by $D$ paths,
and the case where any collection $\mathcal{P}'$ of paths, containing at least one path connecting every demand pair causes congestion $c$, for any values $D$ and $c$ with $Dc = O(\log \log n / \log \log \log n)$, unless $\text{NP} \subseteq \text{ZPTIME}(n^{\text{poly log } n})$. 
CHAPTER 4

GROUP INTEGRAL CONCURRENT FLOW

4.1 Hardness of group-ICF

The goal of this section is to prove Theorem 3. We start by introducing a new scheduling problem, called Max-Min Interval Scheduling (MMIS), and show that it can be cast as a special case of group-ICF. We show later that MMIS is hard to approximate.

4.1.1 Max-Min Interval Scheduling

In the MMIS problem, we are given a collection \( J = \{1, \ldots, n\} \) of \( n \) jobs, and for each job \( j \), we are given a set \( \mathcal{I}_j \) of disjoint closed intervals on the time line. Given any set \( \mathcal{I} \) of time intervals, the congestion of \( \mathcal{I} \) is the maximum, over all time points \( t \), of the number of intervals in \( \mathcal{I} \) containing \( t \). Speaking in terms of scheduling, this is the number of machines needed to execute the jobs during the time intervals in \( \mathcal{I} \). The goal of MMIS is to find, for each job \( j \), a subset \( \mathcal{I}_j^* \subseteq \mathcal{I}_j \) of intervals such that, if we denote \( \mathcal{I}^* = \bigcup_{j=1}^{n} \mathcal{I}_j^* \), then the intervals in \( \mathcal{I}^* \) are disjoint (or equivalently cause congestion 1). The value of the solution is the minimum, over all jobs \( j \in J \), of \( |\mathcal{I}_j^*| \). Given an instance \( J \) of MMIS, we denote by \( N(J) \) the total number of intervals in \( \bigcup_{j \in J} \mathcal{I}_j \). In the following theorem, we relate MMIS to group-ICF.

**Theorem 25** Given any instance \( J \) of MMIS, we can efficiently construct an instance \((G, D)\) of group-ICF on a line graph, such that \( |V(G)| \leq 2N(J) \) and the following holds:

- If \( \text{OPT} \) is the value of the optimal solution to \( J \) with congestion 1, then there is a collection \( \mathcal{P}^* \) of edge-disjoint paths in \( G \), where each pair \( (S_i, T_i) \) is connected by \( \text{OPT} \) paths in \( \mathcal{P}^* \), and

- Given any set \( \mathcal{P}^* \) of paths in \( G \), where every pair \( (S_i, T_i) \) is connected by at least \( D' \)
paths, and the total congestion cased by $\mathcal{P}^*$ is bounded by $c$, we can efficiently find a solution $\mathcal{I}^*$ to instance $\mathcal{J}$ of value $D'$ and congestion $c$.

**Proof:** Given an instance of the MMIS problem, we construct an instance of **group-ICF** on a line graph as follows. Let $\mathcal{J} = \{1, \ldots, n\}$ be the input set of jobs, and let $\mathcal{I} = \bigcup_{j=1}^{n} \mathcal{I}_j$. Let $\mathcal{S}$ be the set of endpoints of all intervals in $\mathcal{I}$. We create a line graph $G = (V,E)$, whose vertex set is $\mathcal{S}$, and the vertices are connected in the order in which they appear on the time line. Clearly, $|V(G)| \leq 2N(\mathcal{J})$. For each job $j$, we create a demand pair $(S_j, T_j)$, as follows. Assume w.l.o.g. that $\mathcal{I}_j = \{I_1, I_2, \ldots, I_r\}$, where the intervals are ordered left-to-right (since all intervals in $\mathcal{I}_j$ are disjoint, this order is well-defined). For each $1 \leq x \leq r$, if $x$ is odd, then we add the left endpoint of $I_x$ to $S_j$ and its right endpoint to $T_j$; otherwise, we add the right endpoint to $S_j$ and the left endpoint to $T_j$. In the end, $|S_j| = |T_j| = |\mathcal{I}_j|$. This concludes the definition of the **group-ICF** instance. Let OPT be the value of the optimal solution to the MMIS instance (with no congestion), and let $\mathcal{I}^* = \bigcup_{j \in \mathcal{J}} \mathcal{I}_j^*$ be the optimal solution to the MMIS problem instance. Notice that for each job $j \in \mathcal{J}$, for each interval $I \in \mathcal{I}_j^*$, one of its endpoints is in $S_j$ and the other is in $T_j$. For each interval $I \in \mathcal{I}^*$, we add the path connecting the endpoints of $I$ to our solution. It is immediate to see that each group $(S_i, T_i)$ is connected by OPT paths, and since intervals in $\mathcal{I}^*$ are disjoint, the congestion is bounded by 1.

Assume now that we are given a solution $\mathcal{P}^* = \bigcup_{j=1}^{n} \mathcal{P}_j^*$ to the **group-ICF** instance, where $\mathcal{P}_j^*$ is the set of paths connecting $S_j$ to $T_j$, such that $|\mathcal{P}_j^*| \geq D'$, and the paths in $\mathcal{P}^*$ cause congestion at most $c$. We now transform $\mathcal{P}^*$ into a solution of value $D'$ and congestion $c$ for the original MMIS instance, as follows.

Consider any path $P$, whose endpoints are $x, y$, where one of these two vertices belongs to $S_j$ and the other to $T_j$. We say that $P$ is **canonical** iff $x, y$ are endpoints of some interval $I \in \mathcal{I}_j$. If some path $P \in \mathcal{P}_j^*$ is not canonical, we can transform it into a canonical path, without increasing the overall congestion, as follows. We claim that path $P$ must contain some interval $I \in \mathcal{I}_j$. Indeed, otherwise, let $S_j$ be the set of endpoints of the intervals in
Then $x$ and $y$ are two consecutive vertices in $S_j$, they are not endpoints of the same interval in $I_j$, and yet one of them belongs to $S_j$ and the other to $T_j$. But the sets $S_j$ and $T_j$ were defined in a way that makes this impossible. Therefore, $P$ contains some interval $I \in I_j$. We truncate path $P$ so that it starts at the left endpoint of $I$ and ends at the right endpoint of $I$. Once we process all paths in $\mathcal{P}^*$, we obtain a solution to the group-ICF problem instance, where all paths are canonical and the congestion is still bounded by $c$. This solution immediately gives us a solution of value $D'$ and congestion at most $c$ to the MMIS problem instance.

We note that a scheduling problem closely related to MMIS is Machine Minimization Job Scheduling, where the goal is to select one time interval for every job, so as to minimize the total congestion. This problem admits an $O(\log n / \log \log n)$-approximation via the Randomized LP-Rounding technique of Raghavan and Thompson [78]. Chuzhoy and Naor [41] have shown that it is $\Omega(\log \log n)$-hard to approximate.

\subsection{4.1.2 Hardness of MMIS}

In the following theorem, we prove hardness of the MMIS problem, which, combined with Theorem 25, immediately implies Theorem 3. Its proof very closely follows the hardness of approximation proof of [41] for Machine Minimization Job Scheduling and appears in the full version of the paper.

**Theorem 26** Suppose we are given an instance $\mathcal{J}$ of MMIS, and let $N = N(\mathcal{J})$. Let $c$ be any integer, $0 < c \leq O(\log \log N)$ and let $D = O \left( N^{1/(2c+3)} \right)$. Then no efficient algorithm can distinguish between the following two cases:

- **(Yes-Instance):** the value of the optimal solution with congestion 1 is at least $D$, and

- **(No-Instance):** any solution $\mathcal{I}^*$, where for all $j \in J$, $|I_j^*| \geq 1$, causes congestion at least $c$. 

76
Combining Theorems 26 and 25 immediately gives the proof of Theorem 3. The proof of Theorem 26 is almost identical to the hardness of approximation proof for Machine Minimization Job Scheduling of [41]. The only difference is that we create more intervals for each job, to ensure that the value of the optimal solution in the Yes-Instance is $D$.

For completeness, we provide a proof here. Our starting point is exactly the same as in [41]: it is the standard 2-prover verifier for the 3SAT(5) problem, whose properties are summarized below.

A 2-prover Verifier

We perform a reduction from the 3SAT(5) problem, in which we are given a 3SAT formula $\varphi$ with $5n/3$ clauses, where each clause contains exactly three distinct literals, and each literal appears in exactly five different clauses. The following statement is equivalent to the PCP theorem [11, 10].

**Theorem 27** There is a constant $\delta > 0$ such that it is NP-hard to distinguish between the formula $\varphi$ that is satisfiable and the one for which any assignment satisfies at most $(1 - \delta)$ fraction of the clauses.

We say that $\varphi$ is a Yes-Instance if it is satisfiable, and if no assignment satisfies a $(1 - \delta)$-fraction of the clauses, we say that it is a No-Instance. We use a standard 2-prover protocol for 3SAT(5), with $\ell$ parallel repetitions. In this protocol, there is a verifier and two provers. Given the input 3SAT(5) formula $\varphi$, the verifier chooses $\ell$ clauses $C_1, \ldots, C_\ell$ of $\varphi$ independently at random, and for each $i : 1 \leq i \leq \ell$, a random variable $x_i$ in clause $C_i$ is chosen. The verifier then sends one query to each one of the two provers. The query to the first prover consists of the indices of clauses $C_1, \ldots, C_\ell$, while the query to the second prover contains the indices of variables $x_1, \ldots, x_\ell$. The first prover is expected to return an assignment to all variables in the clauses $C_1, \ldots, C_\ell$, which must satisfy the clauses, and
the second prover is expected to return an assignment to variables $x_1, \ldots, x_\ell$. Finally, the verifier accepts if and only if the answers from the two provers are consistent. Combining Theorem 27 with the parallel repetition theorem [81], we obtain the following theorem:

**Theorem 28** There is a constant $\gamma > 0$ such that:

- If $\varphi$ is a **Yes-Instance**, then there is a strategy of the two provers such that the acceptance probability of the verifier is 1.

- If $\varphi$ is a **No-Instance**, then for any strategy of the provers, the acceptance probability of the verifier is at most $2^{-\gamma \ell}$.

Let $R$ be the set of all random strings of the verifier, $|R| = (5n)^\ell$. Denote the set of all possible queries to the first and second provers by $Q_1$ and $Q_2$ respectively, so $|Q_1| = (5n/3)^\ell$ and $|Q_2| = n^\ell$. For each query $q \in Q_1 \cup Q_2$, we denote by $A(q)$, the collection of all possible answers to $q$ (if $q \in Q_1$, then we only include answers that satisfy all clauses in $q$). Notice that for each $q \in Q_1$, $|A(q)| = 7^\ell$ and for each $q \in Q_2$, $|A(q)| = 2^\ell$. Given a random string $r \in R$ chosen by the verifier, we let $q_1(r)$ and $q_2(r)$ denote the queries sent to the first and second provers respectively given $r$.

This protocol defines a set $\Phi \subseteq Q_1 \times Q_2$ of constraints, where $(q_1, q_2) \in \Phi$ if and only if for some random string $r \in R$, $q_1 = q_1(r)$ and $q_2 = q_2(r)$. For each constraint $(q_1, q_2) \in \Phi$, we have a corresponding projection $\pi_{q_1,q_2} : A(q_1) \rightarrow A(q_2)$, which specifies the pairs of consistent answers for constraint $(q_1, q_2)$.

We define another set $\Psi \subseteq Q_1 \times Q_1$ of constraints as follows: $(q_1, q'_1) \in \Psi$ if and only if there is a query $q_2 \in Q_2$ such that $(q_1, q_2) \in \Phi$ and $(q'_1, q_2) \in \Phi$. Given a constraint $(q_1, q'_1) \in \Psi$, a pair of answers $a \in A(q_1), a' \in A(q'_1)$ is a satisfying assignment to this constraint iff $\pi_{q_1,q_2}(a) = \pi_{q'_1,q_2}(a')$.

The rest of the reduction consists of two steps. First, we define a basic instance of **MMIS**. We then define our final construction, by combining a number of such basic instances together.
The Basic Instance

In this section, we construct instances of MMIS that are called basic instances, and analyze their properties. A basic instance is determined by several parameters, and the final construction combines a number of basic instances with different parameters.

Let $c$ and $D$ be parameters that we set later. We set the parameter $\ell$ of the Raz verifier to be $\ell = 3c/\gamma$, where $\gamma$ is the constant from Theorem 28. A basic instance is determined by the 3SAT(5) formula $\varphi$ that we reduce from, and the following parameters:

- An integer $k$;
- For each $q \in Q_1$, a collection of $k(q) \leq k$ subsets of assignments $A_1^q, \ldots, A_{k(q)}^q \subseteq A(q)$, where for each $1 \leq i \leq k(q)$, $|A_i^q| \geq |A(q)| - c$.

In order to define the job intervals, we will first define special intervals that we call virtual intervals. Unlike job intervals, virtual intervals are not part of the problem input, but they will be useful in defining the actual job intervals. There are three types of virtual intervals.

1. For each query $q \in Q_1$, we have a virtual interval representing $q$ and denoted by $I(q)$. This interval is called a query interval. All query intervals are equal-sized and non-overlapping.

2. Each query interval $I(q)$ is subdivided into $k(q)$ equal-sized non-overlapping virtual intervals representing the subsets $A_i^q$ for $1 \leq i \leq k(q)$. An interval corresponding to set $A_i^q$ is denoted by $I(A_i^q)$.

3. Finally, each interval $I(A_i^q)$ is subdivided into $|A_i^q|$ non-overlapping virtual intervals, where each such interval represents a distinct assignment $a \in A_i^q$. We denote this interval by $I_i^q(a)$, and we call such intervals assignment intervals. Observe that the same assignment $a$ may appear in several subsets $A_i^q$, and will be represented by a distinct interval for each such subset.
The set $\mathcal{J}$ of jobs for the basic instance is defined as follows. For each constraint $(q, q') \in \Psi$, for each pair $i : 1 \leq i \leq k(q)$, $i' : 1 \leq i' \leq k(q')$ of indices, there is a job $j(A^q_i, A^{q'}_{i'})$ in $\mathcal{J}$ if and only if there is no pair $a, a'$ of assignments such that $a \in A(q) \setminus A^q_i$, $a' \in A(q') \setminus A^{q'}_{i'}$, and $(a, a')$ is a satisfying assignment for the constraint $(q, q')$. If the job $j = j(A^q_i, A^{q'}_{i'})$ exists, we define a set $I_j$ of $D(|A^q_i| + |A^{q'}_{i'}|)$ intervals for $j$, as follows. For each assignment $a \in A^q_i$, we construct $D$ intervals, which are completely contained in $I^q_i(a)$, and add them to $I_j$. Similarly, for each assignment $a' \in A^{q'}_{i'}$, we construct $D$ intervals that are completely contained in $I^{q'}_{i'}(a')$, and add them to $I_j$.

Consider some query $q \in Q_1$, index $1 \leq i \leq k(q)$, and assignment $a \in A^q_i$. Let $\mathcal{I}'$ be the set of indices of all job intervals that need to be contained in $I^q_i(a)$. We then create $|\mathcal{I}'|$ equal-sized non-overlapping sub-intervals of $I^q_i(a)$, and they serve as the intervals corresponding to the indices in $\mathcal{I}'$. This finishes the definition of the basic instance. Notice that all intervals in set $\bigcup_{j \in \mathcal{J}} I_j$ are mutually disjoint. We note that the basic instance is defined exactly like in [41], except that we create $D$ intervals for each job $j$ inside each relevant assignment interval $I^q_i(a)$, while in [41] only one such interval is created. In other words, the construction in [41] is identical to our construction with $D = 1$.

In the next two lemmas, we summarize the properties of the basic instance. The lemmas and their proofs are mostly identical to [41].

**Lemma 9** Assume that $\varphi$ is a Yes-Instance. For each $q \in Q_1$, let $f(q)$ denote the assignment to $q$ obtained from the strategy of the provers, for which the acceptance probability of the verifier is 1. Then we can select, for each job $j \in \mathcal{J}$, a subset $I^*_j \subseteq I_j$ of $D$ intervals, such that only intervals of the form $I^q_i(a)$ for $a = f(q)$ are used in the resulting solution $\bigcup_{j \in \mathcal{J}} I^*_j$.

**Proof:** Consider some job $j = j(A^q_i, A^{q'}_{i'})$. Let $a = f(q)$ and $a' = f(q')$, so $a$ and $a'$ are consistent. Due to the way we define the jobs, either $a \in A^q_i$ or $a' \in A^{q'}_{i'}$. Assume w.l.o.g. that it is the former case. Then we let $I^*_j$ be the set of $D$ intervals of job $j$ that are contained
Let \( I^* \) be any solution to the basic instance. We say that the interval \( I(A^q_i) \) is \textit{used} by \( I^* \), iff there is some job interval \( I \in I^* \), such that \( I \subseteq I(A^q_i) \). We say that a query \( q \in Q \) is \textit{good} iff for all \( i : 1 \leq i \leq k(q) \), interval \( I(A^q_i) \) is used by the solution. The following lemma is identical to Claim 4.6 in [41], and its proof follows from [41].

**Lemma 10** Assume that \( \varphi \) is a \textit{No-Instance}. Then for any solution \( I^* \) that contains at least one interval for each job, at least half of the queries are good.

### 4.2 An Algorithm for group-ICF

We again start with a special case of the problem, where all edge capacities are unit, and all demands are uniform. By appropriately scaling the demands, we can then assume w.l.o.g. that \( \lambda_{\text{opt}} = 1 \), and the demand for each pair \((S_i, T_i)\) is \( D \). Let \( m = \lceil 16 \log n \rceil \) be a parameter. We later define a parameter \( \Delta = k \text{poly log } n \), and we assume throughout the algorithm that:

\[
D \geq 640 \Delta m \alpha_{\text{edg}} \ln^2 n = k \text{poly log } n, \tag{4.1}
\]

by setting \( \Delta' = 640 \Delta m \alpha_{\text{edg}} \ln^2 n \).

We say that a \textit{group-ICF} instance \((G, D)\) is a \textit{canonical instance} iff for all \( 1 \leq i \leq k \), \( S_i = \{s^i_1, \ldots, s^i_{mD}\}, T_i = \{t^i_1, \ldots, t^i_{mD}\} \), and there is a set of paths

\[
\mathcal{P} = \left\{ P^i_j \mid 1 \leq i \leq k, 1 \leq j \leq mD \right\}
\]

where path \( P^i_j \) connects \( s^i_j \) to \( t^i_j \), and the paths in \( \mathcal{P} \) cause congestion at most \( 2m \) in \( G \). We denote by \( \mathcal{M}_i = \left\{ (s^i_j, t^i_j) \right\}_{j=1}^{mD} \) the set of pairs corresponding to \((S_i, T_i)\) for each \( 1 \leq i \leq k \), and we associate a canonical fractional solution \( f^* \) with such an instance, where we send \( 1/(2m) \) flow units along each path \( P^i_j \). The value of such a solution is \( D/2 \) - the total amount of flow sent between each pair \((S_i, T_i)\). Let \( \mathcal{M} = \bigcup_{i=1}^{k} \mathcal{M}_i \). The following lemma allows us
Lemma 11. Given any instance \((G, D)\) of group-ICF with unit edge capacities and uniform demands \(D_i = D\) for all \(1 \leq i \leq k\), where the value of the optimal solution \(\lambda_{opt} = 1\), we can efficiently compute a canonical instance \((G, D', P)\), where for each \(1 \leq i \leq k\), \(|P_i| = Dm\). Moreover, any integral solution to the canonical instance gives an integral solution of the same value and congestion to the original instance.

Proof: Let \((G, D)\) be an instance of group-ICF with uniform demands and unit edge capacities, and let \(f\) be the optimal fractional solution, whose congestion is at most 1. For each \(1 \leq i \leq k\), let \(f_i\) be the flow in \(f\) that originates at the vertices of \(S_i\) and terminates at the vertices of \(T_i\). Recall that \(f_i\) sends \(D\) flow units from the vertices of \(S_i\) to the vertices of \(T_i\).

For each \(1 \leq i \leq k\), we select a set \(P_i = \{P_1^i, \ldots, P_{mD}^i\}\) of \(mD\) flow-paths connecting the vertices in \(S_i\) to the vertices of \(T_i\), as follows. Each path \(P_j^i\), for \(1 \leq j \leq mD\), is selected independently at random, from the set of paths carrying non-zero flow in \(f_i\), where each path \(P\) is selected with probability \(f_i(P)/D\), (here \(f_i(P)\) is the amount of flow on path \(P\)).

Once the set \(P_i\) of paths for each \(1 \leq i \leq k\) is selected, we define a new fractional solution \(f'\), where for each \(1 \leq i \leq k\), we send \(\frac{1}{2m}\) flow units on each path in \(P_i\). It is easy to see that for each \(1 \leq i \leq k\), the amount of flow routed between \(S_i\) and \(T_i\) is exactly \(D/2\). Moreover, using the standard Chernoff bounds, it is easy to see that w.h.p. flow \(f'\) causes congestion at most 1 in \(G\).

From now on, we will assume that the input instance is a canonical one, and that the set \(P\) of paths is given. Throughout this section, the parameter \(L\) in the definition of small and critical clusters is set to \(L = O(\log^{25} n)\), and we set the precise value of \(L\) later.

4.2.1 Split Instances and Good Q-J Decompositions

In this section we introduce two special cases of the group-ICF problem and show efficient algorithms for solving them. In the following section we show an algorithm for the general
problem, which decomposes an input instance of group-ICF into several sub-instances, each of which belongs to one of the two special cases described here.

**Split Instances** The first special case that we define is a split instance. Suppose we are given a canonical instance \((G, D)\) of the group-ICF problem, with the corresponding set \(P\) of paths connecting the demand pairs in \(\bigcup_{i=1}^{k} M_i\). Assume further that we are given a collection \(C = \{C_1, \ldots, C_\ell\}\) of disjoint vertex subsets of \(G\), such that each path \(P \in P\) is completely contained in one of the sets \(C_h\). For each \(1 \leq i \leq k\), for each \(1 \leq h \leq \ell\), let \(P_i(C_h) \subseteq P_i\) be the subset of paths contained in \(C_h\). We say that instance \((G, D)\) is a split instance iff for each \(1 \leq i \leq k\), for each \(1 \leq h \leq \ell\), \(|P_i(C_h)| \leq \frac{D}{4\alpha_{\text{EDP}} \cdot \ln^2 n} = \frac{D}{\text{poly log } n}\).

**Theorem 29** Let \((G, D)\) be a canonical split instance as described above. Then there is an efficient randomized algorithm that finds a collection \(R\) of paths that cause a congestion of at most \(\eta\) in \(G\), and for each \(1 \leq i \leq k\), at least \(\frac{D}{64\alpha_{\text{EDP}} \cdot \ln^2 n} = \frac{D}{\text{poly log } n}\) paths connect the vertices of \(S_i\) to the vertices of \(T_i\) in \(R\) w.h.p.

**Proof:** For each \(1 \leq h \leq \ell\), let \(P(C_h) \subseteq P\) be the subset of paths contained in \(C_h\), and let \(M(C_h) \subseteq M\) be the set of pairs of terminals that these paths connect. Recall that the paths in set \(P(C_h)\) cause a congestion of at most \(2m\) in graph \(G[C_h]\). Therefore, if we route \(1/(2m)\) flow units along each path \(P \in P(C_h)\), then we obtain a feasible fractional solution to the EDP instance on graph \(G[C_h]\) and the set \(M(C_h)\) of demands, where the value of the solution is \(\frac{|M(C_h)|}{2m}\).

Let \(N = 2m\alpha_{\text{EDP}} \cdot \ln n\). We partition the set \(M(C_h)\) into \(N\) subsets \(M_1(C_h), \ldots, M_N(C_h)\), and for each \(1 \leq z \leq N\), we find a collection \(R_z(C_h)\) of paths contained in graph \(G[C_h]\) that connect the demands in \(M_z(C_h)\) and cause congestion at most \(\eta\).

We start with the set \(M(C_h)\) of demands, and apply Theorem 5 to input \((G[C_h], M(C_h))\). Since there is a fractional solution of value \(\frac{|M(C_h)|}{2m}\), we obtain a collection \(R_1(C_h)\) of paths connecting a subset \(M_1(C_h) \subseteq M(C_h)\) of at least \(\frac{|M(C_h)|}{2m\alpha_{\text{EDP}}}\) demand pairs. We then remove the pairs in \(M_1(C_h)\) from \(M(C_h)\) and continue to the next iteration. Clearly,
after $N$ iterations, every pair in the original set $\mathcal{M}(C_h)$ belongs to one of the subsets $\mathcal{M}_1(C_h), \ldots, \mathcal{M}_N(C_h)$. For each such subset $\mathcal{M}_z(C_h)$, we have computed an integral routing $\mathcal{R}_z(C_h)$ of the pairs in $\mathcal{M}_z(C_h)$ inside $G[C_h]$, with congestion at most $\eta_{\text{EDP}}$. We now choose an index $z \in \{1, \ldots, N\}$ uniformly at random, and set $\mathcal{M}'(C_h) = \mathcal{M}_z(C_h)$, and $\mathcal{R}(C_h) = \mathcal{R}_z(C_h)$. We view $\mathcal{R}(C_h)$ as the final routing of pairs in $\mathcal{M}'(C_h)$ inside the cluster $C_h$, and we say that all pairs in $\mathcal{M}'(C_h)$ are routed by this solution. Notice that the probability that a pair in $\mathcal{M}(C_h)$ is routed is $1/N$. The output of the algorithm is $\mathcal{R} = \bigcup_{h=1}^{\ell} \mathcal{R}(C_h)$.

We say that a demand pair $(S_i, T_i)$ is satisfied iff $\mathcal{R}$ contains at least $\frac{mD}{2N} = \frac{D}{4\alpha_{\text{EDP}} \ln n}$ paths connecting the vertices of $S_i$ to the vertices of $T_i$. We now show that w.h.p. every demand pair $(S_i, T_i)$ is satisfied.

Indeed, consider some demand pair $(S_i, T_i)$. Recall that $|\mathcal{P}_i| = Dm$, and for each cluster $C_h \in \mathcal{C}$, $\mathcal{P}_i(C_h) \subseteq \mathcal{P}_i$ is the subset of paths contained in $C_h$. Let $\mathcal{M}_i(C_h) \subseteq \mathcal{M}_i$ be the set of pairs of endpoints of paths in $\mathcal{P}_i(C_h)$. Denote $D_i = \frac{D}{64\alpha_{\text{EDP}} \ln^2 n}$, and recall that we $|\mathcal{M}_i(C_h)| \leq D_i$ must hold. We now define a random variable $y_{i,h}$ as follows. Let $n_{i,h}$ be the number of pairs of vertices in $\mathcal{M}_i(C_h)$ that are routed by $\mathcal{R}(C_h)$. Then $y_{i,h} = \frac{n_{i,h}}{D_i}$. Observe that the variables $\{y_{i,h} \}_{h=1}^{\ell}$ are independent random variables that take values in $[0, 1]$. Let $Y_i = \sum_{h=1}^{\ell} y_{i,h}$. Then the expectation of $Y_i$ is $\mu_i = \frac{mD}{2D_iN} = \frac{64\alpha_{\text{EDP}} \ln^2 n}{2\alpha_{\text{EDP}} \ln n} = 32 \ln n$.

The probability that $(S_i, T_i)$ is not satisfied is the probability that $Y_i < \frac{mD}{2D_iN} = \mu_i/2$. By standard Chernoff bounds, this is bounded by $e^{-\mu_i/8} \leq e^{-4 \ln n} = 1/n^4$. From the union bound, with probability at least $(1 - 1/n^3)$, all pairs $(S_i, T_i)$ are satisfied.

**Good Q-J Decompositions** Suppose we are given a canonical instance $(G, D)$ of the group-ICF problem, and any valid Q-J decomposition $(Q, J)$ of the graph $G$. Recall that for each critical cluster $Q \in \mathcal{Q}$, we are given a partition $\pi(Q)$ of $Q$ into small clusters that have the bandwidth property. Let $\mathcal{X} = J \cup \left( \bigcup_{Q \in \mathcal{Q}} \pi(Q) \right)$. We say that the Q-J decomposition $(Q, J)$ is *good* iff $Q \neq \emptyset$, and no demand pair $(s_j^i, t_j^i)$ is contained in a single cluster in $\mathcal{X}$.

In other words, for each $1 \leq i \leq k$, for each $1 \leq j \leq mD$, vertices $s_j^i$ and $t_j^i$ must belong to
distinct clusters in $X$. We show that if we are given a good $Q$-$J$ decomposition for $G$, then we can efficiently find a good integral solution for it.

For technical reasons that will become apparent later, we state the next theorem for canonical instances with non-uniform demands.

**Theorem 30** Assume that we are given a graph $G$, and for $1 \leq i \leq k$, two collections $S_i = \{s^i_1, \ldots, s^i_{D_i}\}$, $T_i = \{t^i_1, \ldots, t^i_{D_i}\}$ of vertices, where $D_i \geq \Omega \left( L^2 \log^{11} n \right)$. Assume further that we are given a set of paths, $P = \{P^i_j \mid 1 \leq i \leq k, 1 \leq j \leq D_i\}$, where path $P^i_j$ connects $s^i_j$ to $t^i_j$, and the paths in $P$ cause congestion at most $2m$ in $G$. Assume also that we are given a good $Q$-$J$ decomposition $(Q, J)$ for $G$. Then there is an efficient algorithm that finds an integral solution to the group-ICF instance, whose congestion is at most $c_{\text{good}}$, and for each $1 \leq i \leq k$, at least $\left\lfloor \frac{D_i}{\alpha_{\text{good}}} \right\rfloor$ paths connect vertices of $S_i$ to vertices of $T_i$. Here, $\alpha_{\text{good}} = O(\log^{60} n \text{ poly log log } n)$, and $c_{\text{good}}$ is a constant.

**Proof:** Recall that we are given a canonical fractional solution, with a collection $P = \{P^i_j : 1 \leq i \leq k, 1 \leq j \leq D_i\}$ of paths, such that path $P^i_j$ connects $s^i_j$ to $t^i_j$, and no path in $P$ has its two endpoints in the same cluster $X \in \mathcal{X}$, where $\mathcal{X} = J \cup \left( \bigcup_{Q \in Q} \pi(Q) \right)$. We view each path $P^i_j \in P$ as starting at vertex $s^i_j$ and terminating at $t^i_j$. We will repeatedly use the following claim, whose proof follows from standard Chernoff bound, similarly to the proof of Theorem 20.

**Claim 7** Let $P'$ be any collection of paths in $G$, and let $(P'_1, \ldots, P'_k)$ be any partition of $P'$. Assume that we are given another partition $\mathcal{G}$ of the set $P'$ of paths into groups of size at most $q$ each, and assume further that for each $1 \leq i \leq k$, $|P'_i| \geq 32q \log n$. Let $P'' \subseteq P'$ be a subset of paths, obtained by independently selecting, for each group $U \in \mathcal{G}$, a path $P_U \in U$ uniformly at random, so $P'' = \{P_U \mid U \in \mathcal{G}\}$. Then for each $1 \leq i \leq k$, $|P'_i \cap P''| \geq \frac{|P'_i|}{2q}$. 
with high probability.

We say that a path \( P \in \mathcal{P} \) is critical iff it is contained in some critical cluster \( Q \in \mathcal{Q} \). We say that a pair \((S_i, T_i)\) is critical iff the number of paths in \( \mathcal{P}_i \) that are critical is at least \( |\mathcal{P}_i|/2 \). Otherwise, we say that \((S_i, T_i)\) is a regular pair. We first show how to route the critical pairs, and then show an algorithm for routing regular pairs.

**Routing Critical Pairs**  Let \( \hat{\mathcal{P}} \subseteq \mathcal{P} \) be the set of all critical paths, and let \( \mathcal{X}^Q = \bigcup_{Q \in \mathcal{Q}} \pi(Q) \). For each cluster \( X \in \mathcal{X}^Q \), we define two sets of paths: \( U_1(X) \), containing all paths in \( \hat{\mathcal{P}} \) whose first endpoint belongs to \( X \), and \( U_2(X) \), containing all paths in \( \hat{\mathcal{P}} \) whose last endpoint belongs to \( X \). Since cluster \( X \) cannot contain both endpoints of any path in \( \hat{\mathcal{P}} \), and the paths in \( \hat{\mathcal{P}} \) cause congestion at most \( 2m \), while \( |\text{out}(X)| \leq L \), we have that \( |U_1(X)|, |U_2(X)| \leq 2mL \) for all \( X \in \mathcal{X}^Q \). For each cluster \( X \in \mathcal{X}^Q \), we then select, uniformly independently at random, one path \( P_1(X) \in U_1(X) \), and one path \( P_2(X) \in U_2(X) \). We then let \( \hat{\mathcal{P}}' \subseteq \hat{\mathcal{P}} \) be the subset of paths \( P \) that were selected twice in this process. That is, \( \hat{\mathcal{P}}' = \left\{ P_1(X) \mid X \in \mathcal{X}^Q \right\} \cap \left\{ P_2(X) \mid X \in \mathcal{X}^Q \right\} \). Notice that both \( \left\{ U_1(X) \right\}_{X \in \mathcal{X}^Q} \) and \( \left\{ U_2(X) \right\}_{X \in \mathcal{X}^Q} \) are partitions of \( \hat{\mathcal{P}} \) into subsets of size at most \( 2mL \). Therefore, from Claim 7, for each \( 1 \leq i \leq k \), \( |P_i \cap \hat{\mathcal{P}}'| \geq \frac{|P_i|}{16m^2L^2} \) w.h.p.

We now fix some critical cluster \( Q \in \mathcal{Q} \). Let \( \mathcal{P}_Q \subseteq \hat{\mathcal{P}}' \) denote the subset of paths in \( \hat{\mathcal{P}}' \) that are contained in \( Q \), let \( \mathcal{M}_Q \) be the set of pairs of their endpoints, and \( \mathcal{T}_Q \) the subset of terminals that serve as endpoints to the paths in \( \mathcal{P}_Q \). Recall that each small cluster \( C \in \pi(Q) \) contains at most two terminals from \( \mathcal{T}' \). We augment the graph \( G \), by adding, for every terminal \( t \in \mathcal{T}_Q \), an edge \( e_t \) connecting \( t \) to a new vertex \( t' \). Let \( G_Q \) denote this new graph (but note that the new vertices \( t' \) do not belong to \( Q \)). We now show that \( Q \) still has the weight property in this new graph, and each cluster \( C \in \pi(Q) \) still has the bandwidth property (with sligher weaker parameters). We can then use Theorem 10 for routing on critical clusters, to route the pairs in \( \mathcal{M}_Q \).
Claim 8 Each cluster $C \in \pi(Q)$ is $(\alpha_S/3)$-well-linked in graph $G_Q$, and $(Q, \pi(Q))$ has the weight property with parameter $\lambda' = \lambda/3$.

Proof: Consider any small cluster $C \in \pi(Q)$. Cluster $C$ contains at most two endpoints of paths in $P_Q$, so for any subset $A \subseteq C$, $|\text{out}_G(A)| \leq |\text{out}_{G_Q}(A)| \leq |\text{out}_G(A)| + 2$. Since $C$ is $\alpha_S$-well-linked in graph $G$, it is immediate that $C$ is $\alpha_S/3$-well-linked in graph $G_Q$.

Consider now the graph $H_Q$, where we start with $G_Q[|Q|]$, and contract each cluster $C \in \pi$ into a super-node $v_C$, whose original weight $w(v_C)$ is $|\text{out}_G(C)|$, and the new weight $w'(v_C)$ is $|\text{out}_{G_Q}(C)|$. Notice that $w(v_C) \leq w'(v_C) \leq w(v_C) + 2 \leq 3w(v_C)$. Since $(Q, \pi(Q))$ has the weight property with parameter $\lambda$ in graph $G$, for any partition $(A, B)$ of $V(H_Q)$, $|E_{H_Q}(A, B)| \geq \lambda \min \left\{ \sum_{v \in A} w(v), \sum_{v \in B} w(v) \right\} \geq \frac{\lambda}{3} \min \left\{ \sum_{v \in A} w'(v), \sum_{v \in B} w'(v) \right\}.$

Therefore, $(Q, \pi(Q))$ has the weight property with parameter $\lambda' = \lambda/3$. □

We can now use Theorem 10 to find a grouping $G_Q$ of the terminals in $T_Q$ into groups of size $O(Z) = O(\log^4 n)$, such that for any set $D$ of $(1, G_Q)$-restricted demands, there is an efficient randomized algorithm that w.h.p. routes $D$ integrally in $G[|Q|]$ with constant congestion. Grouping $G_Q$ defines two partitions $G_Q^1, G_Q^2$ of the paths in $P_Q$, as follows: for each group $U \in G_Q$, we have a subset $P_1(U) \subseteq P_Q$ of paths whose first endpoint belongs to $U$, and a subset $P_2(U) \subseteq P_Q$ of paths whose last endpoint belongs to $U$. We let $G_Q^1 = \{ P_1(U) \mid U \in G_Q \}$, and $G_Q^2 = \{ P_2(U) \mid U \in G_Q \}$. For each group $U \in G$, we randomly sample one path in $P_1(U)$ and one path in $P_2(U)$. We let $P'_Q$ be the set of all paths that have been selected twice, once via their first endpoint and once via their last endpoint, and we let $\hat{P}' = \bigcup_{Q \in Q} P'_Q$. From Claim 7, for each critical $(S_i, T_i)$ pair, $|P_i \cap \hat{P}'| \geq \Omega \left( \frac{|P_i|}{m^2L^2Z^2} \right) = \Omega \left( \frac{|P_i|}{L^2 \log^{4} n} \right)$ w.h.p. For each $Q \in Q$, let $M'_Q$ be the set of pairs of endpoints of paths in $P'_Q$. Then $M'_Q$ defines a set of $(2, G_Q)$-restricted demands on the set $T_Q$ of terminals, and from Theorem 10, there is a randomized algorithm to route
these demands in $G[Q]$ with constant congestion.

**Routing Regular Pairs** We use the graph $H$, given by Theorem 15. The algorithm consists of two steps. In the first step, we route some of the terminals to the boundaries of the critical clusters, and create what we call “fake terminals”. This step is very similar to the algorithm of Andrews [5]. In this way, we transform the problem of routing the original terminal pairs in graph $G$ into a problem of routing the new fake terminal pairs in graph $H$. The second step is very similar to the proof of Theorem 21: we split graph $H$ into $x = \text{poly log } n$ sub-graphs $H_1, \ldots, H_x$ using standard edge sampling, and route a subset of the fake demand pairs in each graph $H_j$ using the algorithm of Rao and Zhou [79].

Since we now focus on regular pairs only, to simplify notation, we assume that we are given a collection $\{(S_i, T_i)\}_{i=1}^k$ of regular demand pairs, and a collection $\mathcal{P} = \{P^i_j\}_{1 \leq i \leq k, 1 \leq j \leq D'_i}$ of paths, where $D'_i = D_i/2$, such that path $P^i_j$ connects $s^i_j$ to $t^i_j$. We assume that all paths $P^i_j$ are regular. For each $1 \leq i \leq k$, $S_i = \{s^i_1, \ldots, s^i_{D'_i}\}$, and $T_i = \{t^i_1, \ldots, t^i_{D'_i}\}$. Let $\mathcal{T} = \bigcup_{i=1}^k (S_i \cup T_i)$ be the set of all terminals, and for $1 \leq i \leq k$, let $\mathcal{M}_i = \{(s^i_j, t^i_j)\}_{j=1}^{D'_i}$ be the set of the pairs of endpoints of paths in set $\mathcal{P}_i = \{P^i_1, \ldots, P^i_{D'_i}\}$. Denote by $\mathcal{T}^J$ and $\mathcal{T}^Q$ the sets of all terminals contained in the $J$- and the $Q$-clusters respectively, that is, $\mathcal{T}^J = \mathcal{T} \cap (\bigcup_{C \in J} C)$, and $\mathcal{T}^Q = \mathcal{T} \cap (\bigcup_{C \in Q} C)$.

**Step 1: Defining fake terminal pairs** The goal of this step is to define new pairs of terminals, that we call fake terminal pairs, in graph $H$, so that a good routing of the fake terminal pairs in graph $H$ immediately translates into a good routing of the original pairs in graph $G$. This step largely follows the ideas of [5]. Let $E^Q = \bigcup_{Q \in \mathcal{Q}} \text{out}(Q)$. Our first step is to route all terminals in $\mathcal{T}$ to the edges of $E^Q$. This is done using the following three lemmas.

**Lemma 12** There is an efficient algorithm to find a partition $\mathcal{G}^J$ of the set $\mathcal{T}^J$ of terminals into groups of size at most $48m$, such that for any $(2, \mathcal{G}^J)$-restricted subset $\mathcal{T}' \subseteq \mathcal{T}^J$ of
terminals, there is an efficient algorithm to find a set $P_J : \mathcal{T}' \leadsto_{12} E^Q$ of paths in $G$.

**Proof:** Let $E^J = \bigcup_{J \in \mathcal{J}} \text{out}(J)$. Since no pair of endpoints of paths in $\mathcal{P}$ is contained in a single cluster $J \in \mathcal{J}$, there is a set $P_1 : \mathcal{T}' \leadsto_{4m} E^J$ of paths, connecting every terminal $t \in \mathcal{T}'$ to some edge in $E^J$. The collection $P_1$ of paths is obtained as follows: let $t \in \mathcal{T}$ be the endpoint of some path $P \in \mathcal{P}$, and assume that $t \in C$, where $C$ is a $J$-cluster. Let $e \in \text{out}(C)$ be the first edge on path $P$ that is not contained in $C$. We then add the segment of $P$ between $t$ and $e$ to the set $P_1$ of paths. This defines the set $P_1 : \mathcal{T}' \leadsto_{4m} E^J$ of paths.

Recall that we are also given a set $N$ of paths called tendrils, $N : E^J \leadsto_{3} E^Q$. Combining the two sets of paths, we obtain a collection $P_2 : \mathcal{T}' \leadsto_{16m} E^Q$.

We now use the standard grouping technique, to partition the terminals in $\mathcal{T}'$ into groups of size at least $16m$ and at most $48m$. Let $\mathcal{G}^J$ be this resulting grouping. Since each group can simultaneously send one flow unit to the edges of $E^Q$ with congestion 1, it is immediate to see that for any $(1, \mathcal{G}^J)$-restricted subset $\mathcal{T}' \subseteq \mathcal{T}'$ of terminals, there is a flow $F_J : \mathcal{T}' \leadsto_2 E^Q$. From the integrality of flow, there must be a set $P_1' : \mathcal{T}' \leadsto_{2} E^Q$ of paths, which can be computed efficiently. Finally, using Observation 1, we conclude that for any $(2, \mathcal{G}^J)$-restricted subset $\mathcal{T}' \subseteq \mathcal{T}'$ of terminals, we can efficiently find a set $P_J : \mathcal{T}' \leadsto_{12} E^Q$ of paths.

**Lemma 13** We can efficiently find a partition $\mathcal{G}^Q$ of the set $\mathcal{T}^Q$ of terminals into groups of size at most $48m$, such that for any $(2, \mathcal{G}^Q)$-restricted subset $\mathcal{T}' \subseteq \mathcal{T}^Q$ of terminals, there is an efficient algorithm to find a set $P_Q : \mathcal{T}' \leadsto_{12} E^Q$ of paths $G$.

**Proof:** We start by showing that there is a collection $P_2 : \mathcal{T}^Q \leadsto_{2m} E^Q$ of paths in $G$, which is constructed as follows. Let $t \in \mathcal{T}$ be the endpoint of some path $P \in \mathcal{P}$, and assume that $t \in C$, where $C$ is a $Q$-cluster. Since $P$ is a regular path, it is not completely contained in $C$. Let $e \in \text{out}(C)$ be the first edge on path $P$ that is not contained in $C$. We then add the segment of $P$ between $t$ and $e$ to the set $P_2$ of paths. Since the paths in $\mathcal{P}$ cause congestion at most $2m$, the total congestion due to paths in $P_2$ is bounded by $4m$. 

89
The rest of the proof is identical to the proof of Lemma 12: we compute a grouping $G^Q$ of the terminals in $T^Q$ into groups of size at least $4m$ and at most $12m$, using the standard grouping technique. It is then immediate to see that for any for any $(1, G^Q)$-restricted subset $T' \subseteq T^Q$ of terminals, there is a flow $F^Q : T' \sim_{12} E^Q$, and hence a set $P^Q : T' \sim_{12} E^Q$ of paths. Using Observation 1, we conclude that for any $(2, G^Q)$-restricted subset $T' \subseteq T^Q$ of terminals, we can efficiently find a set $P^Q : T' \sim_{12} E^Q$ of paths. \hfill $\square$

Let $G = G^Q \cup G^J$ be the partition of terminals in $T$ obtained from Lemmas 12 and 13. For each group $U \in G$ of terminals, we define two subsets of paths: $P_1(U) \subseteq P$ contains all paths whose first endpoint belongs to $U$, and $P_2(U) \subseteq P$ contains all paths whose last endpoint belongs to $U$. We then select, independently uniformly at random, two paths $P_1(U) \in P_1(U)$ and $P_2(U) \in P_2(U)$. Let $P' \subseteq P$ be the subset of paths that have been selected twice, that is, $P' = \{P_1(U) \mid U \in G\} \cap \{P_2(U) \mid U \in G\}$. Since both $\{P_1(U)\}_{U \in G}$ and $\{P_2(U)\}_{U \in G}$ define partitions of the paths in $P$ into sets of size at most $48m$, from Claim 7, for each $1 \leq i \leq k$, $|P_i \cap P'| \geq \frac{|P_i|}{4608m^2}$ w.h.p.

Let $T' \subseteq T$ be the set of terminals that serve as endpoints for paths in $P'$. Since set $T'$ is $(2, G)$-restricted, from Lemmas 12 and 13, there is a collection $R : T' \sim_{24} E^Q$ of paths in graph $G$. For each terminal $t \in T'$, set $R$ contains a path $P_t$, connecting $t$ to some edge $e_t \in E^Q$. We define a mapping $f : T' \rightarrow E^Q$, where $f(t) = e_t$.

Recall that for each critical cluster $Q \in Q$, Theorem 10 gives a partition $G(Q)$ of the edges of $\operatorname{out}(Q)$ into subsets of size at most $3Z = O(\log^4 n)$.

Consider some edge $e \in E^Q$. If there is a single critical cluster $Q \in Q$ such that $e \in \operatorname{out}(Q)$, then we say that $e$ belongs to $Q$. Otherwise, if there are two such clusters $Q_1, Q_2 \in Q$, then we select one of them arbitrarily, say $Q_1$, and we say that $e$ belongs to $Q_1$. We will view $G(Q)$ as a partition of only those edges in $\operatorname{out}(Q)$ which belong to $Q$, and we will ignore all other edges. Let $G' = \bigcup_{Q \in Q} G(Q)$, so $G'$ is a partition of $E^Q$. Our final step is to sample the paths in $P'$, such that for each group $U \in G'$, there are at most two paths whose endpoints are mapped to the edges of $U$. 

90
For each group $U \in \mathcal{G}'$, we define a subset $\mathcal{P}_1(U) \subseteq \mathcal{P}'$ of paths, containing all paths whose first endpoint $t$ is mapped to an edge of $U$, that is, $f(t) \in U$, and similarly, a subset $\mathcal{P}_2(U) \subseteq \mathcal{P}'$ of paths, containing all paths whose last endpoint is mapped to an edge of $U$. We then select, uniformly independently at random, a path $P_1(U) \in \mathcal{P}_1(U)$, and a path $P_2(U) \in \mathcal{P}_2(U)$, and let $\mathcal{P}'' \subseteq \mathcal{P}'$ be the subset of paths that have been selected twice, that is, $\{P_1(U) \mid U \in \mathcal{G}\} \cap \{P_2(U) \mid U \in \mathcal{G}\}$. Since each set $|\mathcal{P}_1(U)|, |\mathcal{P}_2(U)| \leq 3Z = O(\log^4 n)$, from Claim 7, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \mathcal{P}''| \geq \Omega \left( \frac{D_i}{m^2 Z^2} \right) = \Omega \left( \frac{D_i}{\log^{10} n} \right)$ w.h.p.

Let $\mathcal{T}'' \subseteq \mathcal{T}$ be the set of terminals that serve as endpoints of paths in $\mathcal{P}''$, and let $\mathcal{R}'' \subseteq \mathcal{R}$ be their corresponding subset of paths, $\mathcal{R}'' : \mathcal{T}'' \leadsto_{24} E^Q$. For each $1 \leq i \leq k$, let $\mathcal{P}''_i = \mathcal{P}_i \cap \mathcal{P}''$, and let $\mathcal{M}''_i$ be the set of pairs of endpoints of the paths in $\mathcal{P}''_i$.

Let $H$ be the graph given by Theorem 15. We are now ready to define fake demand pairs for the graph $H$. For each $1 \leq i \leq k$, we define a set $\tilde{\mathcal{M}}_i$ of demand pairs, and the sets $\tilde{\mathcal{S}}_i, \tilde{\mathcal{T}}_i$ of fake terminals, as follows: for each pair $(s, t) \in \mathcal{M}''_i$, if $Q_1 \in \mathcal{Q}$ is the critical cluster to which $f(s)$ belongs, and $Q_2 \in \mathcal{Q}$ the critical cluster to which $f(t)$ belongs, then we add $(v_{Q_1}, v_{Q_2})$ to $\tilde{\mathcal{M}}_i$, and we add $v_{Q_1}$ to $\tilde{\mathcal{S}}_i$ and $v_{Q_2}$ to $\tilde{\mathcal{T}}_i$. Notice that we allow $\tilde{\mathcal{M}}_i, \tilde{\mathcal{S}}_i$ and $\tilde{\mathcal{T}}_i$ to be multi-sets. This finishes the definition of the fake demand pairs. In order to complete Step 1, we show that any good integral solution to this new group-ICF instance will give a good integral solution to the original group-ICF instance.

**Lemma 14** Let $\tilde{\mathcal{P}}$ be any collection of paths in graph $H$ that causes congestion at most $\gamma$. For each $1 \leq i \leq k$, let $n_i$ be the number of paths in $\tilde{\mathcal{P}}$ connecting the fake terminals in $\tilde{\mathcal{S}}_i$ to the fake terminals in $\tilde{\mathcal{T}}_i$. Then we can efficiently find a collection $\mathcal{P}^*$ of paths in the original graph $G$, such that for each $1 \leq i \leq k$, there are $n_i$ paths connecting the terminals of $\tilde{\mathcal{S}}_i$ to the terminals of $\tilde{\mathcal{T}}_i$ in $\mathcal{P}^*$, and the congestion caused by paths in $\mathcal{P}^*$ is at most $c_0 \gamma$ for some constant $c_0$.

**Proof:** Consider some path $\tilde{P} \in \tilde{\mathcal{P}}$, and let $(v_{Q_1}, v_{Q_2}) \in \tilde{\mathcal{M}}$ be the endpoints of path $\tilde{P}$. Let $(s, t) \in \mathcal{M}$ be the original pair of terminals that defined the pair $(v_{Q_1}, v_{Q_2})$ of fake
terminals. We transform the path ˜\(P\) into a path \(P\) connecting \(s\) to \(t\) in graph \(G\). In order to perform the transformation, we start with the path ˜\(P\), and replace its endpoints and edges with paths in graph \(G\). Specifically, we replace \(v_{Q_1}\) with the path \(P_s \in \mathcal{R}''\) that connects \(s\) to some edge in \(\text{out}(Q_1)\), and we replace \(v_{Q_2}\) with the path \(P_t \in \mathcal{R}''\), connecting some edge in \(\text{out}(Q_2)\) to \(t\). Additionally, for each edge \(e = (v_Q, v_{Q'})\) on path \(P\), we replace \(e\) by the path \(P_e\) connecting some edge in \(\text{out}(Q)\) to some edge in \(\text{out}(Q')\), given by Theorem 15. So far, each path ˜\(P\) is replaced by a sequence of paths (that we call segments) in graph \(G\). For each pair \(\sigma, \sigma'\) of consecutive segments, there is a critical cluster \(Q \in \mathcal{Q}\), such that the last edge of \(\sigma\) and the first edge of \(\sigma'\) belong to \(\text{out}(Q)\). For each critical cluster \(Q \in \mathcal{Q}\), we now define a set \(\mathcal{D}(Q)\) of demands on the edges of \(\text{out}(Q)\), as follows: for each path ˜\(P\) \(\in \tilde{\mathcal{P}}\), for each pair \((\sigma, \sigma')\) of consecutive segments that we have defined for path ˜\(P\), where the last edge \(e\) of \(\sigma\), and the first edge \(e'\) of \(\sigma'\) belong to \(\text{out}(Q)\), we add the demand pair \((e, e')\) to \(\mathcal{D}(Q)\). We allow \(\mathcal{D}(Q)\) to be a multi-set.

Since the paths in ˜\(P\) cause congestion at most \(\gamma\), from Property C4, and from the fact that the terminals in \(\mathcal{T}''\) are \((2, \mathcal{G})\)-restricted, we get that for each critical cluster \(Q \in \mathcal{Q}\), the set \(\mathcal{D}(Q)\) of demands is \((2\gamma + 2)\)-restricted. Combining Observation 1 with Theorem 10, we get that the set \(\mathcal{D}(Q)\) of demands can be routed inside \(Q\) with congestion at most \(547 \cdot (6\gamma + 6)\).

For each path ˜\(P\) \(\in \tilde{\mathcal{P}}\), we now combine the segments we have defined for ˜\(P\) with the routings we have computed inside the critical clusters to connect these segments, to obtain the final path \(P\) in graph \(G\).

\[\square\]

**Step 2: Routing in graph \(H\)** In this step, we find a solution for the group-ICF problem defined on graph \(H\) and the set of fake terminal pairs. For each \(1 \leq i \leq k\), let \(\tilde{D}_i = |\tilde{\mathcal{M}}_i|\), and recall that \(\tilde{D}_i = \Omega\left(\frac{D_i}{\log^{\Omega} n}\right)\). For each \(1 \leq i \leq k\), we will route a polylogarithmic fraction of the demand pairs in \(\tilde{\mathcal{M}}_i\), with no congestion in graph \(H\). This step is almost identical to the proof of Theorem 21, except that we use different parameters. For simplicity, we assume w.l.o.g. in this step that all values \(\tilde{D}_i\) are equal. In order to achieve this, let \(D\) be
the minimum value of \( \tilde{D}_i \) over all \( 1 \leq i \leq k \). For each \( 1 \leq i \leq k \), we partition the demand pairs in \( \tilde{\mathcal{M}}_i \) into subsets, containing \( D \) pairs each (except possibly the last subset). If one of the resulting subsets contains fewer than \( D \) pairs, we simply disregard all pairs in this subset. In this way, we define a new collection of demand pairs \( \{ \tilde{\mathcal{M}}_{i'} \} \), where \( 1 \leq i' \leq k' \). Notice that it is now enough to find a collection \( \tilde{\mathcal{P}} \) of paths, such that for each new group \( \tilde{\mathcal{M}}_{i'} \), at least a poly-logarithmic fraction of the pairs in \( \tilde{\mathcal{M}}_{i'} \) are connected by paths in \( \tilde{\mathcal{P}} \).

To simplify notation, we now assume w.l.o.g. that for each \( 1 \leq i \leq k \), \( \tilde{D}_i = D \).

Let \( \alpha_{rz} = O(\log^{10} n) \), \( L_{rz} = \Theta(\log^5 n) \) be the parameters from Theorem 23. Let \( x = 16m\alpha_{rz} \cdot \log n = O(\log^{12} n) \). We set \( L = 2\alpha^* \cdot x \cdot L_{rz} = O(\log^{25} n \text{ poly log log } n) \).

We split graph \( H \) into \( x \) graphs \( H_1, \ldots, H_x \), as follows. For each \( 1 \leq j \leq x \), we have \( V(H_j) = V(H) \). In order to define the edge sets of graphs \( H_j \), each edge \( e \in E \), chooses an index \( 1 \leq j \leq x \) independently uniformly at random, and is then added to \( E(H_j) \). This completes the definition of the graphs \( H_j \).

For convenience, we define a new graph \( G' \), which is obtained from the original graph \( G \) by contracting each cluster \( Q \in \mathcal{Q} \) into a super-node \( v_Q \). Notice that the set \( \tilde{\mathcal{M}} \) of fake terminal pairs is also a set of pairs of vertices in graph \( G' \), so we can also view \( \tilde{\mathcal{M}} \) as defining a set of demand pairs in graph \( G' \).

Given any partition \( (A, B) \) of the vertices of \( V(H) \), let \( \text{cut}_{G'}(A, B) \) denote the value of the minimum cut \( |E_{G'}(A', B')| \) in graph \( G' \), such that \( A \subseteq A', B \subseteq B' \). Theorem 15 guarantees that the size of the minimum cut in \( H \) is at least \( L/\alpha^* \), and for each partition \( (A, B) \) of \( V(H) \), \( \text{cut}_{G'}(A, B) \leq \alpha^* \cdot |E_H(A, B)| \). From Theorem 16, for each graph \( H_j \), for \( 1 \leq j \leq x \), w.h.p. we have that: (i) The value of the minimum cut in \( H_j \) is at least \( \frac{L}{2\alpha^* \cdot x} = L_{rz} \); and (ii) For any partition \( (A, B) \) of \( V(H_j) \), \( |E_{H_j}(A, B)| \geq \frac{\text{cut}_{G'}(A, B)}{2\alpha^*} \).

We need the following lemma.

**Lemma 15** For each \( 1 \leq j \leq x \), there is a fractional solution to the instance \( (H_j, \tilde{\mathcal{M}}) \) of group-ICF, where each demand pair in \( \tilde{\mathcal{M}} \) sends \( \frac{1}{6m \alpha^* \cdot \Delta_{FCG}} \) flow units to each other with no congestion.
Proof: Assume otherwise. Then the value of the maximum concurrent flow in graph $H_i$ for the set $\mathcal{M}$ of demands is less than $\frac{1}{6mn\alpha^*\beta_{\text{FCG}}}$. We set up an instance of the non-uniform sparsest cut problem on graph $H_j$ with the set $\tilde{\mathcal{M}}$ of demand pairs. Then there is a cut $(A, B)$ in graph $H_j$, with $\frac{|E_{H_j}(A,B)|}{D_{H_j}(A,B)} < \frac{1}{6mn\alpha^*}$. Let $(A', B')$ be the minimum cut in graph $G'$, where $A \subseteq A', B \subseteq B'$. Then $|E_{G'}(A', B')| = \text{cut}_{G'}(A, B) \leq 2\alpha^*|E_{H_j}(A, B)|$, while $D_{G'}(A', B') = D_{H_j}(A, B)$. Therefore, $\frac{|E_{G'}(A', B')|}{D_{G'}(A', B')} \leq 2\alpha^* \frac{|E_{H_j}(A, B)|}{D_{H_j}(A, B)} < \frac{1}{3m}$. We next show that there is a concurrent flow, in graph $G'$, of value $\frac{1}{3m}$ and no congestion, between the pairs of the fake terminals in $\tilde{\mathcal{M}}$. This contradicts the fact that the value of the sparsest cut in graph $G'$ is less than $\frac{1}{3m}$.

In order to find the concurrent flow in graph $G'$, it is enough to show that, for every pair $(v_Q, v_{Q'}) \in \tilde{\mathcal{M}}$ of fake terminals, we can send one flow unit from some vertex of $Q$ to some vertex of $Q'$ simultaneously with congestion at most $3m$ in graph $G$. Scaling this flow down by factor $3m$ will define the desired flow in graph $G'$.

Consider some pair $(v_Q, v_{Q'}) \in \tilde{\mathcal{M}}$ of fake terminals and let $(s, t) \in \mathcal{M}$ be the original demand pair that defined the pair $(v_Q, v_{Q'})$. Recall that there is a path $P \in \mathcal{P}$ connecting $s$ to $t$, and paths $P_s, P_t \in \mathcal{R}''$, where $P_s$ connects $s$ to some vertex $u \in Q$, and $P_t$ connects $t$ to some vertex $u' \in Q'$. The concatenation of these three paths gives a path that connects $u$ to $u'$ in graph $G$. Since the paths in $\mathcal{P}$ cause congestion at most $2m$, while the paths in $\mathcal{R}''$ cause congestion at most 24, the total congestion caused by these flow-paths is at most $2m + 24 < 3m$. We conclude that there is a concurrent flow in graph $G'$ of value $\frac{1}{3m}$ between the pairs of terminals in $\tilde{\mathcal{M}}$, contradicting our former conclusion that the value of the sparsest cut in $G'$ is less than $\frac{1}{3m}$.

In the rest of the algorithm, we apply the algorithm of Rao-Zhou to each of the graphs $H_1, \ldots, H_x$ in turn, together with some subset $\tilde{\mathcal{M}} \subseteq \tilde{\mathcal{M}}$ of the fake demand pairs. The output of the iteration is a collection $\tilde{\mathcal{P}}^j$ of edge-disjoint paths in graph $H_j$ connecting some demand pairs in $\tilde{\mathcal{M}}^j$. We say that a pair $(\tilde{S}_i, \tilde{T}_i)$ is satisfied in iteration $j$, iff $\tilde{\mathcal{P}}^j$ contains at least $\frac{D}{48m\alpha^*\alpha_{\text{RG}}}$ paths connecting demand pairs in $\tilde{\mathcal{M}}_i$. 

94
We now fix some $1 \leq j \leq x$ and describe the execution of iteration $j$. Let $I \subseteq \{1, \ldots, k\}$ be the set of indices $i$, such that $(\tilde{S}_i, \tilde{T}_i)$ was not satisfied in iterations $1, \ldots, j - 1$. Let $\tilde{\mathcal{M}}' = \bigcup_{i \in I} \tilde{\mathcal{M}}_i$. From Lemma 15, there is a fractional solution $F$ in graph $H_j$, where every demand pair in $\tilde{\mathcal{M}}'$ sends $\frac{1}{6mx^\alpha \beta_{\text{FCG}}}$ flow units to each other with no congestion. We transform instance $(H_j, \tilde{\mathcal{M}}')$ of group-ICF into a canonical instance, obtaining, for each $i \in I$, a collection $\tilde{\mathcal{P}}'_i$ of $\lfloor \frac{D}{6x^\alpha \beta_{\text{FCG}}} \rfloor$ paths, such that the set $\bigcup_{i \in I} \tilde{\mathcal{P}}'_i$ of paths causes congestion at most $2m$ in graph $H_j$. Let $\tilde{\mathcal{M}}^j$ be the collection of pairs of endpoints of paths in $\bigcup_{i \in I} \tilde{\mathcal{P}}'_i$. We apply Theorem 23 to the EDP instance defined by the graph $H_j$ and the set $\tilde{\mathcal{M}}^j$ of the demand pairs. Let $\tilde{\mathcal{P}}^j$ be the output of the algorithm. Then w.h.p., $|\tilde{\mathcal{P}}^j| \geq \frac{|\tilde{\mathcal{M}}^j|}{2ma_{\text{az}}}$, and the paths in $\tilde{\mathcal{P}}^j$ are edge-disjoint.

It is easy to verify that at least $\frac{1}{8ma_{\text{az}}}$-fraction of pairs $(\tilde{S}_i, \tilde{T}_i)$ for $i \in I$ become satisfied in iteration $j$. This is since $|\tilde{\mathcal{P}}^j| \geq \frac{|\tilde{\mathcal{M}}^j|}{2ma_{\text{az}}} \geq \frac{|I| \cdot D}{24ma_{\text{az}} x^\alpha \beta_{\text{FCG}}}$, each unsatisfied pair contributes at most $\frac{D}{48ma_{\text{az}} x^\alpha \beta_{\text{FCG}}}$ paths to $\tilde{\mathcal{P}}^j$, and each satisfied pair contributes at most $\frac{D}{6x^\alpha \beta_{\text{FCG}}}$ paths. Therefore, after $x = 16ma_{\text{az}} \cdot \log n$ iterations, all demand pairs are satisfied.

Let $\tilde{\mathcal{P}} = \bigcup_{j=1}^x \tilde{\mathcal{P}}^j$ denote the final collection of paths. For each demand pair $(\tilde{S}_i, \tilde{T}_i)$, set $\tilde{\mathcal{P}}$ must contain at least $\frac{\tilde{D}_i}{48mx^\alpha \beta_{\text{FCG}} \cdot a_{\text{az}}} = \Omega \left( \frac{D_i}{\log^2 n \cdot \text{poly} \log \log n} \right)$ paths connecting the vertices of $\tilde{S}_i$ to the vertices of $\tilde{T}_i$, and the paths in $\tilde{\mathcal{P}}$ are edge-disjoint. Applying Lemma 14, we obtain a collection $\mathcal{P}^*$ of paths in graph $G$, that cause a constant congestion, and for each $1 \leq i \leq k$, at least $\Omega \left( \frac{D_i}{\log^2 n \cdot \text{poly} \log \log n} \right)$ paths connect the vertices of $S_i$ to the vertices of $T_i$.

\subsection{The Algorithm}

We now present an algorithm to solve a general canonical instance. Our algorithm uses the following parameter:
\[
\Delta = \max \begin{cases} 
640kmL_{\text{ARS}}(n)\log n \\
\alpha_{\text{EDP}} \\
16mL_{\text{good}}c_{\text{good}} \\
\Omega(L^2\log^{11}n)
\end{cases}
\]

which gives \( \Delta = k \poly \log n \). Let \( \mathcal{T} = \bigcup_{i=1}^{k} (S_i \cup T_i) \) be the set of all terminals. The main idea is that we would like to find a \( Q-J \) decomposition of the graph \( G \), such that each small cluster \( X \in \mathcal{X} \), where \( \mathcal{X} = \mathcal{J} \cup \left( \bigcup_{Q \in \mathcal{Q}} \pi(Q) \right) \), contains at most \( D/\poly \log n \) terminals. Suppose we can find such a decomposition. We then say that a path \( P \in \mathcal{P} \) is of type 1 iff its both endpoints are contained in some set \( X \in \mathcal{X} \), and it is of type 2 otherwise. We can then partition the demand pairs \((S_i, T_i)\) into two types: a demand pair \((S_i, T_i)\) is of type 1 if most of the paths in \( \mathcal{P}_i \) are of type 1, and it is of type 2 otherwise. This partitions the problem instance into two sub-instances: one induced by type-1 demand pairs, and the other induced by type-2 demand pairs. The former is a split instance w.r.t. \( \mathcal{X} \), while for the latter instance, the current \( Q-J \) decomposition is a good decomposition. We can then solve the two resulting sub-instances using Theorems 29 and 30, respectively.

We are however unable to find such a \( Q-J \) decomposition directly. Instead, our algorithm consists of three steps. In the first step, we find a partition of \( V(G) \) into subsets \( V_1, \ldots, V_r \), and for each \( 1 \leq h \leq r \), we let \( G_h = G[V_h] \). This partition guarantees that for each resulting graph \( G_h \), for each small cut \((A, B)\) in graph \( G_h \), either \( A \) or \( B \) contain at most \( \Delta \) terminals. Moreover, the number of edges \( e \in E(G) \) whose endpoints do not lie in the same set \( V_h \) is at most \( D/4 \). This partition decomposes our original problem into \( r \) sub-problems, where for \( 1 \leq h \leq r \), the \( h \)th subproblem is defined over the graph \( G_h \). In the second step, for each graph \( G_h \), we find a \( Q-J \) decomposition \( (Q_h, \mathcal{J}_h) \). Let \( \mathcal{X}_h = \mathcal{J} \cup \{ \pi(Q) \mid Q \in Q_h \} \) be the corresponding set of small clusters. While the \( Q-J \) decomposition is not necessarily good, we ensure that each cluster \( C \in \mathcal{X}_h \) may only contain a small number of pairs \((s^i_j, t^i_j)\) for all
\[1 \leq i \leq k.\] We then decompose the demand pairs \((S_i, T_i)\) into several types, and define a separate sub-instance for each of these types. We will ensure that each one of the resulting instances is either a split instance, or the \(Q-J\) decomposition computed in step 2 is a good decomposition for it.

**Step 1: Partitioning the graph \(G\)** This step is summarized in the following lemma, whose proof is similar to the proof of standard well-linked decomposition.

**Lemma 16** Let \((G, D)\) be a canonical instance of group-ICF with uniform demands. Then there is an efficient algorithm to find a partition \(R\) of \(V\), such that for each \(R \in R\), for any partition \((A, B)\) of \(R\), where \(|E_G[R](A, B)| < 2L\), either \(A\) or \(B\) contain at most \(\Delta\) terminals. Moreover, \(\sum_{R \in R} |\text{out}(R)| \leq D/4\).

**Proof:**

The proof is similar to standard well-linked decomposition procedures, except that we use slightly different parameters. Throughout the algorithm, we maintain a partition \(R\) of \(V\), where at the beginning, \(R = \{V\}\).

Consider some set \(R \in R\). Let \(G' = G[R]\), and let \(T_R = T \cap R\). Let \((A, B)\) be any partition of \(R\), denote \(T_A = A \cap T\), \(T_B = B \cap T\), and assume w.l.o.g. that \(|T_A| \leq |T_B|\). We say that the cut \((A, B)\) is sparse iff \(|E_{G'}(A, B)| \leq \frac{2L \cdot \alpha_{\text{ARV}}(n)}{\Delta} \cdot |T_A|\). We run algorithm \(\mathcal{A}_{\text{ARV}}\) on instance \((G', T_R)\) of the sparsest cut problem. Let \((A, B)\) be the output of this algorithm. If the cut \((A, B)\) is sparse, then we remove \(R\) from \(R\), and we add \(A\) and \(B\) instead. Assume w.l.o.g. that \(|T_A| \leq |T_B|\). Then we charge the edges in \(T_A\) evenly for the edges in \(|E_{G'}(A, B)|\). Notice that the charge to every edge is at most \(\frac{2L \cdot \alpha_{\text{ARV}}(n)}{\Delta}\).

If algorithm \(\mathcal{A}_{\text{ARV}}\) returns, for each \(R \in R\), a cut that is not sparse, then we stop the execution of the algorithm, and return the set \(R\). Notice that we are now guaranteed that for each set \(R \in R\), if \((A, B)\) is any partition of \(R\), \(T_A = A \cap T\), \(T_B = B \cap T\), and \(|T_A| \leq |T_B|\), then \(|E_{G[R]}(A, B)| \geq \frac{2L}{\Delta} \cdot |T_A|\). In particular, if \(|E_{G[R]}(A, B)| < 2L\), then \(|T_A| < \Delta\) must hold.
In order to bound $\sum_{R \in \mathcal{R}} |\text{out}(R)|$, consider some edge $e = (u, v)$. In each iteration that $e$ is charged via the vertex $u$, the charge is at most $\frac{2L_{ARV}(n)}{\Delta}$, and the number of terminals in the cluster to which $e$ belongs goes down by the factor of at least 2. Therefore, the total direct charge to $e$ via $u$ is at most $\frac{2L_{ARV}(n) \log n}{\Delta}$, and the total direct charge to $e$ via both endpoints is bounded by $\frac{4L_{ARV}(n) \log n}{\Delta} \leq \frac{1}{32 km}$ since $\Delta \geq 128 km L_{ARV}(n) \log n$. The indirect charge forms a geometric series, so overall we can bound: $\sum_{R \in \mathcal{R}} |\text{out}(R)| \leq |\mathcal{T}| \frac{8}{32} \leq \frac{D}{32}$.

We apply Lemma 16 to our instance $(G, D)$, to obtain a partition $\mathcal{R} = \{V_1, \ldots, V_r\}$ of $V(G)$. We denote $E' = \bigcup_{R \in \mathcal{R}} \text{out}(R)$, and for each $1 \leq h \leq r$, we denote $G_h = G[V_h]$. Consider now some demand pair $(S_i, T_i)$. Since $|E'| \leq D/4$, and the set $\mathcal{P}_i$ of paths causes congestion at most $2m$ in $G$, there are at least $mD/2$ pairs $(s^i_j, t^i_j) \in \mathcal{M}_i$, for which the path $P^i_j$ is completely contained in some graph $G_h$. Let $\mathcal{M}'_i \subseteq \mathcal{M}_i$ be the subset of all such pairs $(s^i_j, t^i_j)$, $|\mathcal{M}'_i| \geq mD/2$, and let $\mathcal{P}'_i \subseteq \mathcal{P}_i$ be the subset of their corresponding paths.

For each $1 \leq h \leq r$, let $\mathcal{M}_{i,h} \subseteq \mathcal{M}'_i$ be the subset of pairs $(s^i_j, t^i_j)$ for which $P^i_j$ is contained in $G_h$, let $\mathcal{P}_{i,h} \subseteq \mathcal{P}'_i$ be the subset of paths connecting such pairs, and let $D_{i,h} = |\mathcal{P}_{i,h}|$. Notice that $\sum_{h=1}^r |\mathcal{P}_{i,h}| \geq Dm/2$. For each $1 \leq h \leq r$, let $\mathcal{P}^h = \bigcup_{i=1}^k \mathcal{P}_{i,h}$, and let $f^h$ be the fractional solution associated with the set $\mathcal{P}^h$ of paths, where each path in $\mathcal{P}^h$ is assigned $1/(2m)$ flow units. Then for each $1 \leq i \leq k$, flow $f^h$ routes $D_{i,h}/(2m)$ flow units between the pairs in $\mathcal{M}_{i,h}$, and the flow $f^h$ causes no congestion in $G^h$. We let $\mathcal{T}^h$ be the set of all terminals participating in pairs in $\bigcup_{i=1}^k \mathcal{M}_{i,h}$.

**Step 2: constructing Q-J decompositions** We say that a graph $G_h$ is small iff $|\mathcal{T}^h| \leq 8\Delta$, and otherwise we say that it is large. The goal of this step is to find a Q-J decomposition for each large graph $G_h$. In this step we fix some graph $G' = G_h$, where $G_h$ is a large graph, and we focus on finding a Q-J decomposition for it. We denote $\mathcal{T}' = \mathcal{T}^h$ to simplify notation. Recall that $|\mathcal{T}'| > 8\Delta$.

Suppose we are given a Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of $G'$, and let $\mathcal{X} = \mathcal{J} \cup (\bigcup_{C \in \mathcal{Q}} \pi(C))$. We say that this decomposition is non-trivial iff $\mathcal{Q} \neq \emptyset$. We say that it is successful iff each
cluster $X \in \mathcal{X}$ contains at most $\Delta$ terminals in $\mathcal{T}'$. Notice that in general, Lemma 16 ensures that every cluster $X \in \mathcal{X}$ contains either at most $\Delta$ or at least $|\mathcal{T}'| - \Delta$ terminals from $\mathcal{T}'$, since for each $X \in \mathcal{X}$, $|\text{out}_{\mathcal{T}'}(X)| \leq L$. In order for a decomposition to be successful, we need to ensure that the latter case never happens.

We will instead achieve a decomposition with slightly weaker properties. Let $S^* \subseteq V(G')$ be any vertex subset containing at most $\Delta$ terminals, with $|\text{out}_{\mathcal{T}'}(S^*)| \leq 2L$ (we do not require that $G'[S^*]$ is connected). Let $G_{S^*}$ be the graph obtained from $G'$ as follows: if $|\text{out}_{\mathcal{T}'}(S^*)| \leq L$, then we set $G_{S^*} = G' \setminus S^*$. Otherwise, $L < |\text{out}_{\mathcal{T}'}(S^*)| \leq 2L$, and we obtain $G_{S^*}$ from $G'$ by contracting the vertices of $S^*$ into a super-node $v_{S^*}$. In this step, we show an efficient algorithm to find a set $S^*$ with $|\text{out}_{\mathcal{T}'}(S^*)| \leq 2L$, together with a successful non-trivial $Q$-$J$ decomposition for the graph $G_{S^*}$. The algorithm is summarized in the next theorem, whose proof appears in Section 4.2.3.

**Theorem 31** There is an efficient algorithm to find a cluster $S^* \subseteq V(G')$ containing at most $\Delta$ terminals, with $|\text{out}_{\mathcal{T}'}(S^*)| \leq 2L$, and a successful non-trivial $Q$-$J$ decomposition for the graph $G_{S^*}$.

For each graph $G_h$, if $G_h$ is large, we invoke Theorem 31 to obtain set $S^*_h$ and $(\mathcal{Q}_h, \mathcal{J}_h)$, a $Q$-$J$ decomposition of graph $G_{S^*_h}$. We denote $\mathcal{X}_h = \mathcal{J}_h \cup \left( \bigcup_{Q \in \mathcal{Q}_h} \pi(Q) \right)$, and $\mathcal{X}'_h = \mathcal{X}_h \cup \{S^*_h\}$. Otherwise, if $G_h$ is small, then we denote $\mathcal{X}'_h = \{V(G_h)\}$. Finally, let $\mathcal{X} = \bigcup_{h=1}^r \mathcal{X}'_h$.

Consider some path $P \in \mathcal{P}'$. We say that this path is of type 1 iff it is completely contained in some cluster $X \in \mathcal{X}$. Assume now that the endpoints of $P$ are contained in some cluster $X \in \mathcal{X}$, but $P$ is not completely contained in cluster $X$. If $X = S^*_h$ for some $1 \leq h \leq r$, then we say that $P$ is of type 2; otherwise, it is of type 3. All remaining paths are of type 4.

We partition the demand pairs $(S_i, T_i)$ into four types. We say that a demand pair $(S_i, T_i)$ is of type 1, iff at least $1/5$ of the paths in $\mathcal{P}'_i$ are of type 1; we say that it is of type 2 iff at least $1/5$ of the paths in $\mathcal{P}'_i$ are of type 2; similarly, it is of type 3 iff at least $1/5$ of
the paths in $\mathcal{P}_i'$ are of type 3, and otherwise it is of type 4. If a pair belongs to several types, we select one of the types for it arbitrarily.

**Step 3: Routing the demands** We route the demands of each one of the four types separately.

**Type-1 demands** It is easy to see that type-1 demands, together with the collection $\mathcal{X}$ of clusters, define a split instance. This is since each cluster $X \in \mathcal{X}$ contains at most $\Delta$ demand pairs, and $\Delta \leq \frac{D}{640m_\text{EDP} \cdot \log^2 n}$ from Equation (4.1). If $(S_i, T_i)$ is a type-1 demand, then the number of type-1 paths in $\mathcal{P}_i'$ is at least $Dm/10$. Therefore, we can apply Theorem 29, and obtain a collection $\mathcal{R}^1$ of paths that cause congestion at most $\eta_\text{EDP}$ in $G$, and for each type-1 pair $(S_i, T_i)$, at least $D/\text{poly log } n$ paths connect the vertices in $S_i$ to the vertices in $T_i$ in $R^1$ w.h.p.

**Type-2 Demands** We show that the set of type-2 demands, together with the collection of vertex subsets $V_h$ where $G_h$ is large, define a valid split instance. Indeed, for each such subset $V_h$ of vertices, every type-2 path that is contained in $V_h$ must contain an edge in $\text{out}_{G_h}(S^*_h)$. Since there are at most $2L$ such edges, and the paths in $\mathcal{P}$ cause congestion at most $2m$, we get that the number of type-2 paths contained in each such subset $V_h$ is bounded by $4mL < \Delta \leq \frac{D}{640m_\text{EDP} \cdot \log^2 n}$. For each type-2 demand pair $(S_i, T_i)$, there are at least $\frac{Dm}{10}$ type-2 paths connecting the vertices of $S_i$ to the vertices of $T_i$ in $\mathcal{P}_i'$. Therefore, we can apply Theorem 29, and obtain a collection $\mathcal{R}^2$ of paths that cause congestion at most $\eta_\text{EDP}$ in $G$, and for each type-2 demand pair $(S_i, T_i)$, at least $D/\text{poly log } n$ paths connect the vertices in $S_i$ to the vertices in $T_i$ in $\mathcal{R}^2$ w.h.p.

**Type-3 Demands** Let $X \in \mathcal{X} \setminus \{S^*_1, \ldots, S^*_r\}$, and consider the set $\mathcal{P}(X)$ of type-3 paths whose both endpoints belong to $X$. Assume w.l.o.g. that $X \subseteq V_h$. Since $|\text{out}_{G_h}(X)| \leq 2L$, $|\mathcal{P}(X)| \leq 4Lm$ must hold. Recall that $G_h[X]$ is a connected graph if $X \not\in \{S^*_1, \ldots, S^*_r\}$. 100
Let $\mathcal{M}(X)$ be the collection of pairs of endpoints of the paths in $\mathcal{P}(X)$. We select one pair $(s, t) \in \mathcal{M}(X)$ uniformly at random, and we connect $s$ to $t$ by any path contained in $G[X]$. Let $\mathcal{R}^3$ be the set of all such resulting paths. Using the same arguments as in Theorem 29, it is easy to see that w.h.p. every type-3 demand pair $(S_i, T_i)$ has at least $D/\text{poly log } n$ paths connecting the vertices of $S_i$ to the vertices of $T_i$ in $\mathcal{R}^3$, since $4mL < \frac{D}{16 \log^* n}$.

**Type-4 Demands** Let $(S_i, T_i)$ be any type-4 demand, and let $\mathcal{P}_i^4 \subseteq \mathcal{P}_i'$ be the subset of type-4 paths for $(S_i, T_i)$. Recall that $|\mathcal{P}_i^4| \geq Dm/5$. For each $1 \leq h \leq r$, let $\mathcal{P}_i^4(h) \subseteq \mathcal{P}_i^4$ be the subset of paths contained in the graph $G_h$. We say that pair $(S_i, T_i)$ is light for $G_h$ iff

$$|\mathcal{P}_i^4(h)| < \max\left\{16mL\alpha_{\text{good}}c_{\text{good}}, \Omega(L^2 \log^{11} n)\right\}.$$

Otherwise, we say that it is heavy for $G_h$. We say that a demand pair $(S_i, T_i)$ is light iff the total number of paths in sets $\mathcal{P}_i^4(h)$, where $(S_i, T_i)$ is light for $G_h$ is at least $Dm/10$. Otherwise, we say that it is heavy.

Let $(S_i, T_i)$ be any demand pair, and let $P$ be any type-4 path connecting a vertex of $S_i$ to a vertex of $T_i$. Assume w.l.o.g. that $P$ is contained in $G_h$ for some $1 \leq h \leq r$. We say that $P$ is a light path if $(S_i, T_i)$ is light for $G_h$, and we say that it is a heavy path otherwise.

We now construct two canonical instances. The first instance consists of light $(S_i, T_i)$ demand pairs of type 4, and their corresponding light type-4 paths in $\mathcal{P}_i'$. It is easy to see that this defines a split instance for the collection of vertex subsets $V_h$, where $G_h$ is large. This is since for each light pair $(S_i, T_i)$, for each subset $V_h$ where $(S_i, T_i)$ is light for $G_h$, $|\mathcal{P}_i^4(h)| < \max\left\{16mL\alpha_{\text{good}}c_{\text{good}}, \Omega(L^2 \log^{11} n)\right\} \leq \Delta \leq \frac{D}{640ma_{\text{EHP}} \ln^2 n}$. Therefore, we can use Theorem 29 to find a collection $\mathcal{R}_4$ of paths that cause congestion at most $\eta_{\text{EHP}}$, and for each light type-4 pair $(S_i, T_i)$, at least $D/\text{poly log } n$ paths connect the vertices of $S_i$ to the vertices of $T_i$ in $\mathcal{R}_4$ w.h.p.

Finally, consider some heavy type-4 pair $(S_i, T_i)$. Let $\mathcal{P}_i'' \subseteq \mathcal{P}_i'$ be the set of all heavy type-4 paths in $\mathcal{P}_i'$, and let $\mathcal{M}_i''$ be the set of pairs of their endpoints. Recall that $|\mathcal{M}_i''| \geq Dm/10,$
and the paths in $P''_i$ cause congestion at most $2m$ in $G$. For each $1 \leq h \leq r$, where $G_h$ is large, let $P''_i(h) \subseteq P''_i$ be the subset of paths contained in $G_h$, and let $M''_i(h)$ be the set of their endpoints.

Consider some large graph $G_h$, and consider the sets $(S'_i, T'_i)$ of demands for $1 \leq i \leq k$, where $S'_i$ contains the first endpoint and $T'_i$ contains the last endpoint of every path in $P''_i(h)$. Then the $Q$-J decomposition that we have computed in Step 2 is a good decomposition for graph $G'_h$, where $G'_h$ is obtained from $G_h$ by contracting the vertices of $S'_h$ into a supernode $v_h$, and it is possible that some paths in $R^5(h)$ contain the vertex $v_h$. However, since the degree of $v_h$ is bounded by $2L$, and the congestion due to paths in $R^5(h)$ is at most $c_{\text{good}}$, there are at most $2Lc_{\text{good}}$ such paths in $R^5(h)$. We simply remove all such paths from $R^5(h)$. Since for each $1 \leq i \leq k'$, $R^5(h)$ contains more than $4Lc_{\text{good}}$ paths connecting $S_i$ to $T_i$, we delete at most half the paths connecting each pair $(S_i, T_i)$ in set $R^5(h)$.

Let $R^5 = \bigcup_h R^5(h)$. Then for each heavy type-4 pair $(S_i, T_i)$, at least $\frac{Dm}{4lm\alpha_{\text{good}}} = \frac{D}{\text{poly log } n}$ paths connect $S_i$ to $T_i$ in $R^5$, since we are guaranteed that for all $h$, either $D_i(h) = 0$, or $D_i(h) \geq 2m\alpha_{\text{good}}$. The congestion due to paths in $R^5$ is bounded by $c_{\text{good}}$.

Our final solution is $P^* = \bigcup_{j=1}^5 R_j$. From the above discussion, for every pair $(S_i, T_i)$, set $P^*$ contains at least $D/\text{poly log } n$ paths connecting $S_i$ to $T_i$, and the congestion due to $P^*$ is bounded by a constant.

### 4.2.3 Proof of Theorem 31

Our first step is to find an initial critical cluster $Q_0$ in graph $G'$, that will be used to initialize the set $Q_0$, that we use as input to Theorem 14 in order to find a non-trivial $Q$-J decomposition. We start by finding some large cluster $S$ in graph $G'$ that has the

102
bandwidth property and then use Lemma 4 to find a critical cluster $Q_0 \subseteq S$. The next claim is formulated for a slightly more general setting, in which we will use it later.

**Claim 9** Let $G$ be any graph, and $\tilde{T}$ be any set of terminals in $G$ with $|\tilde{T}| \geq 4\Delta$, such that for any partition $(A, B)$ of the vertices of $G$ with $|E_G(A, B)| \leq L$, either $A$ or $B$ contain at most $\Delta$ terminals. Then there is an efficient algorithm to find a large cluster $S$ in $G$ that has the bandwidth property.

**Proof:** We say that a cut $(A, B)$ of $V(G)$ is balanced, iff $|\tilde{T} \cap A|, |\tilde{T} \cap B| > \Delta$. We start with some arbitrary balanced cut $(A, B)$, and assume w.l.o.g. that $|\tilde{T} \cap A| \leq |\tilde{T} \cap B|$. We then perform a number of iterations. In each iteration, we start with a balanced cut $(A, B)$, where $|\tilde{T} \cap A| \leq |\tilde{T} \cap B|$. At the end of the iteration, we either declare that $B$ is a large cluster with the bandwidth property, or find another balanced cut $(A', B')$ with $|E_G(A', B')| < |E_G(A, B)|$. Therefore, after $|E(G)|$ iterations, the algorithm must terminate with a large cluster that has the bandwidth property.

We now describe an execution of an iteration. We start with a balanced cut $(A, B)$. Denote $T_A = \tilde{T} \cap A$ and $T_B = \tilde{T} \cap B$, and recall that $|T_A| \leq |T_B|$. Since the cut is balanced, and $|T_A|, |T_B| > \Delta, |E_G(A, B)| \geq L$ must hold, so $B$ is a large cluster.

We next run the algorithm $A_{ARV}$ on the instance of the sparsest cut problem defined by the graph $G[B]$, where the set of the terminals is the edges of out$_G(B)$. If the output is a cut whose sparsity is greater than $\frac{1}{2}$, then we are guaranteed that $B$ has the bandwidth property, so we output $B$. Assume now that the algorithm produces a cut $(X, Y)$ whose sparsity is less than $\frac{1}{2}$. Let $T_X = \tilde{T} \cap X$ and $T_Y = \tilde{T} \cap Y$, and assume w.l.o.g. that $|T_X| \leq |T_Y|$. Let $(A', B')$ be a new partition of $V(G)$, where $A' = A \cup X$, and $B' = Y$. It is easy to see that $(A', B')$ remains a balanced cut, since $|\tilde{T}| \geq 4\Delta$. It now only remains to show that $|E_G(A', B')| < |E_G(A, B)|$.

Indeed, $|E_G(A', B')| = |E_G(X, Y)| + |E_G(A, Y)|$ which is less than $\frac{|\text{out}_G(X)\text{out}_G(B)|}{2} + |E_G(A, Y)| < |E_G(A, X)| + |E_G(A, Y)| = |E_G(A, B)|$. \qed
CHAPTER 5
EXTENSIONS, OTHER RESULTS AND CONCLUDING REMARKS

5.1 Arbitrary Edge Capacities and Demands

In this section we extend our algorithms for basic-ICF and for group-ICF from Sections ?? and 4.2 to arbitrary demands and edge capacities. We only present here the generalization for basic-ICF, since the extension of the algorithm for group-ICF to general edge capacities and demands is almost identical.

Let $\alpha = \text{poly log } n$ denote the approximation factor of the algorithm from Section ??, and let $\gamma$ denote the congestion. Recall that for each demand pair $(t, t') \in \mathcal{M}$, the algorithm finds $\lfloor \lambda_{\text{OPT}} D/\alpha \rfloor$ paths connecting $t$ to $t'$ in $G$.

We now assume that we are given a set $\mathcal{D}$ of arbitrary demands, and the edge capacities are also arbitrary. We assume w.l.o.g. that $\lambda_{\text{OPT}} = 1$. Let $D_{\text{max}}$ and $D_{\text{min}}$ be the maximum and the minimum demands in $\mathcal{D}$, respectively. We first consider the case where $D_{\text{max}}/D_{\text{min}} \leq n^3$, and show a factor $4\alpha$-approximation with congestion at most $\gamma$ for it.

If $D_{\text{min}} \geq 2n^3$, then we delete all edges whose capacities are less than $\rho = \lfloor D_{\text{min}}/(2n^3) \rfloor$. Notice that the total amount of flow going through such edges in the optimal fractional solution is bounded by $D_{\text{min}}/2$, so this change reduces the value of the optimal fractional solution by at most factor 2. We then divide all demands and edge capacities by the factor $\rho$, thus obtaining a new problem instance $G'$. The value of the optimal solution for $G'$ remains $\lambda_{\text{OPT}} = 1$, and any integral solution of value $\lambda$ and congestion $\eta$ in $G'$ can be converted into an integral solution of value $\lambda$ and congestion $\eta$ in $G$. From now on we will assume w.l.o.g. that the value of the minimum demand $D_{\text{min}} \leq 4n^3$, and so the value of the maximum demand, $D_{\text{max}} \leq 4n^6$, while $\lambda_{\text{OPT}} = 1$.

Let $D^* = 4\alpha/\lambda_{\text{OPT}}$. Since we are only interested in finding a factor $4\alpha$-approximation, we can assume w.l.o.g., that for each pair of terminals, either $D(t, t') = 0$, or $D(t, t') \geq D^*$. 104
In particular, $D_{\text{min}} \geq D^*$. We now slightly modify the graph $G$, and define a new set $\mathcal{D}'$ of demands, such that in the new instance all capacities are unit, and the demands are uniform. Let $D = D_{\text{min}}$. We start with $\mathcal{M} = \emptyset$. Consider some demand pair $(s, t)$ with $D(s, t) > 0$, and let $N(s, t) = \lfloor D(s, t)/D \rfloor$. We create $N(s, t)$ copies of the source $s$ that connect to $s$ with a capacity-$\infty$ edge each, and $N(s, t)$ copies of the sink $t$ that connect to $t$ with capacity-$\infty$ edges. We also add $N(s, t)$ disjoint pairs of vertices, each containing one copy of $s$ and one copy of $t$, to set $\mathcal{M}$. Let $\mathcal{M}$ be the final set of terminal pairs, obtained after we process all pairs with non-zero demands, and let $\mathcal{D}'$ be the corresponding set of demands, where for each pair $(s', t') \in \mathcal{M}$, we set its demand $D'(s', t')$ to be $D$. Notice that so far for each pair $(s, t)$ of vertices with $D(s, t) > 0$, the total demand in $\mathcal{D}'$ between the copies of $s$ and the copies of $t$ is at least $D(s, t)/2$ and at most $D(s, t)$. Therefore, an $\alpha'$-approximate solution to the resulting instance will give a $2\alpha'$-approximation to the original instance. Our final step is to take care of the non-uniform edge capacities. Since we are interested in finding an integral routing, we can assume w.l.o.g. that all edge capacities $c_e \geq 1$.

Since $\lambda_{\text{opt}} = 1$, the total flow through any edge in the optimal fractional solution cannot exceed $n^2 \cdot D_{\text{max}}$, so if the capacity of any edge is greater than $n^2 \cdot D_{\text{max}}$, we can set it to $n^2 \cdot D_{\text{max}}$ without changing the value of the optimal fractional solution. Finally, for each edge $e$, we replace $e$ with $\lceil c(e) \rceil$ parallel edges with unit capacities. The resulting instance of ICF has unit edge capacities and uniform demands. The value of the optimal fractional solution is at least $\lambda_{\text{opt}}/2$, where $\lambda_{\text{opt}}$ is the value of the optimal fractional solution in the original instance. We can now use the algorithm from Section ?? to find an $\alpha$-approximate integral solution with congestion at most $\gamma$ for this new instance. This solution immediately gives a factor $4\alpha$-approximation with congestion at most $2\gamma$ for the original instance.

Assume now that we are given an instance $(G, \mathcal{D})$ of basic-ICF with arbitrary demands and capacities. By appropriately scaling the demands, we can assume w.l.o.g. that $\lambda_{\text{opt}} = 1$. We group the demands geometrically into groups $\mathcal{D}_1, \mathcal{D}_2, \ldots$, where group $\mathcal{D}_i$ contains all
demands $D(s, t)$ with $n^{3(i-1)}D'_{\text{min}} \leq D(s, t) < n^{3i}D'_{\text{min}}$, where $D'_{\text{min}}$ is the minimum demand in $\mathcal{D}$. Notice that the number of non-empty groups $\mathcal{D}_i$ is bounded by $n^2$. For each non-empty group $\mathcal{D}_i$, we create a new instance of $\text{basic-ICF}$, as follows. We build a graph $G_i$ whose set of vertices is $V(G)$, and the set of edges consists of all edges of $G$ whose capacities are at least $n^{3(i-2)}D'_{\text{min}}$. If the capacity of an edge is more than $n^{3i+2}D'_{\text{min}}$, then we set its capacity to $n^{3i+2}$. The capacities of all other edges remain unchanged. We then use the algorithm for the special case where $D_{\text{max}}/D_{\text{min}} \leq n^3$ for each one of the resulting instances $(G_i, \mathcal{D}_i)$, and output the union of their solutions. Since the value of the optimal fractional solution in each such instance is at least $\lambda_{\text{OPT}}/2$, it is immediate to verify that we obtain a factor $8\alpha$-approximation. In order to bound the edge congestion, observe that for each edge $e \in E(G)$, the total capacity of copies of edge $e$ in all instances $(G_i, \mathcal{D}_i)$ to which $e$ belongs is bounded by $4c(e)$. Therefore, the overall edge congestion is bounded by $8\gamma$.

5.2 Gaps Between Fractional and Integral group-ICF

In this section, we show tight bounds on the integrality gap of LP relaxation for group-ICF. We start by showing an integrality gap lower bound of $n^{1/c}$ when congestion $c$ is allowed, and we provide an algorithm achieving this bound. We remark, again, that this bound only holds when $D << k$.

5.2.1 Lower Bound

We start by showing that if no congestion is allowed, then the ratio between $\lambda_{\text{OPT}}$ - the optimal fractional solution and $\lambda^*$ - the optimal integral solution can be as large as $\Omega(\sqrt{n})$ for group-ICF, even if $k = 2$.

Our gap example is a slight modification of the $n \times n$ grid. We start from a grid $G$ with $V(G) = \{v(i, j) | (i, j) \in [D] \times [D]\}$, where $v(i, j)$ denotes the vertex on the $i$th row and $j$th column. We add new sets $S_1 = \{s_1, \ldots, s_D\}$, $T_1 = \{t_1, \ldots, t_D\}$, $S_2 = \{s'_1, \ldots, s'_D\}$ and
$T_2 = \{t'_1, \ldots, t'_D\}$ of vertices (see Figure 5.1). For each $1 \leq j \leq D$, we connect $s_j$ to $v(j, 1)$, $v(j, D)$ to $t_j$, $v(1, j)$ to $s'_j$, and $v(D, j)$ to $t'_j$. Finally, for each $i, j \in [D]$, we replace vertex $v(i, j)$ by a gadget that is shown in the figure. Denote the resulting graph by $G'$.

![Figure 5.1: The gap example construction](image)

The fractional solution can send $D/2$ flow units from $S_i$ to $T_i$ for $i \in \{1, 2\}$ in $G'$, where for each $j$, $s_j$ sends a $\frac{1}{2}$-flow unit to $t_j$ using the paths corresponding to the horizontal grid lines, and $s'_j$ sends a $\frac{1}{2}$-flow unit to $t'_j$, using the paths corresponding to the vertical grid lines. It is easy to see that the value of the integral solution is 0: Assume that an integral solution $P$ contains a path $P_1$ connecting a vertex of $S_1$ to a vertex of $T_1$, and a path $P_2$ connecting a vertex of $S_2$ to a vertex of $T_2$. Consider the corresponding paths $P'_1, P'_2$ in the $D \times D$ grid. Then paths $P'_1$ and $P'_2$ must cross at some vertex $v(i, j)$ of the grid. But then $P_1$ and $P_2$ must share an edge from the gadget corresponding to $v(i, j)$. Since the number of vertices in $G'$ is $n = O(D^2)$, this shows a gap of $\Omega(\sqrt{n})$ between the optimal fractional and the optimal integral solutions.

Next, we show a gap example when congestion is allowed. Let $c$ be a constant and $D$
be a parameter. We show an instance where we can fractionally send $D = \Theta(n^{1/(2c)})$ flow units between each pair $(S_i, T_i)$, while any integral solution containing at least one path for each $(S_i, T_i)$ will cause congestion at least $2c$.

Our construction is iterative. An input to iteration $j$ is a partition $\mathcal{I}_{j-1}$ of the interval $[0, 1]$ of the real line into $(2Dc)^{j-1}$ sub-intervals, that we call level-$(j-1)$ intervals. At the beginning, $\mathcal{I}_0$ consists of a single level-0 interval $[0, 1]$. An iteration is executed as follows. Let $I \in \mathcal{I}_{j-1}$ be any level-$(j-1)$ interval, and let $\ell, r$ be its left and right endpoints, respectively. We subdivide interval $I$ into $2D$ equal-length segments by adding $2D + 1$ new terminals $\{s_1(I), t_1(I), s_2(I), t_2(I), \ldots, s_{cD}(I), t_{cD}(I), s_{cD+1}(I)\}$, corresponding to points on the real line, where $\ell = s_1(I) < t_1(I) < \ldots < s_{cD}(I) < t_{cD}(I) = r$. We then define a new demand $(S(I), T(I))$, where $S(I) = \{s_h(I)\}_{h=1}^{cD+1}$, $T(I) = \{t_h(I)\}_{h=1}^{cD}$. Each of the new sub-intervals of $I$ becomes a level-$j$ interval, and serves as the input to the next iteration.

The final construction is the line graph $G$ obtained after $2c$ iterations. Notice that $|V(G)| = (2cD)^{2c} + 1$, and the number of the demand pairs is $k = \sum_{j=0}^{2c-1} (2cD)^j = \Theta((2cD)^{2c-1})$.

In the fractional solution, for each $1 \leq j \leq 2c$, for each level-$(j-1)$ interval $I$, each sink terminal $t_h(I)$ receives $1/(2c)$ flow units from each of the two source terminals to its right and to its left. Thus, we send $\frac{1}{2c} \cdot 2cD = D$ flow units between the vertices of $S(I)$ and the vertices of $T(I)$. The congestion of this fractional solution is 1, since each edge is contained in at most one interval per level, and we have $2c$ levels.

Consider now any integral solution that contains at least one path connecting every demand pair $(S(I), T(I))$. We show that the congestion of this solution is $2c$. Let $I_0$ be the unique level-0 interval $[0, 1]$, and consider the path $P_1$ connecting some terminal in $S(I_0)$ to some terminal in $T(I_0)$. Then path $P$ must contain some level-1 interval $I_1$. We then consider the path $P_2$ connecting some terminal of $S(I_1)$ to some terminal of $T(I_1)$. This path in turn must contain some level-2 interval $I_2$. We continue like this until we reach level $(2c)$, thus constructing a collection $I_1, \ldots, I_{2c}$ of nested intervals. Each such interval is contained
in a distinct path that the solution uses. Therefore, the congestion of the solution is at least 2c.

### 5.2.2 Upper Bound

Given a fractional solution of value $\lambda_{\text{OPT}}$, we show an algorithm that finds $\tilde{\Omega}(\lambda_{\text{OPT}} D/n^{1/c})$ paths that cause congestion at most $c$. This bound matches the upper bound in the previous section, up to logarithmic factor.

Let $D' = mD\lambda_{\text{OPT}}$. Similarly to Section 4.2, we transform a fractional solution into a canonical solution of the following form. We have $S_i = \{s^i_1, \ldots, s^i_{mD'}\}$, $T_i = \{t^i_1, \ldots, t^i_{mD'}\}$, and there is a set of paths $P = \{P^i_j : 1 \leq i \leq k, 1 \leq j \leq mD'\}$, where $P^i_j$ connects $s^i_j$ to $t^i_j$, and the paths in $P$ cause congestion at most $2m$ in $G$. We denote by $P_i$ the set of paths connecting some vertex in $S_i$ to some vertex in $T_i$.

We now describe the algorithm. For each $i : 1 \leq i \leq k$, we partition $P_i$ into $\left\lfloor \frac{D'}{gm} \right\rfloor$ subsets of size $gm$ each. For each such subset, we select one path uniformly at random. Let $P' \subseteq P$ be the set of selected paths, so it is clear that $P'$ contains at least $\left\lfloor \frac{D\lambda_{\text{OPT}}}{g} \right\rfloor$ paths connecting some vertex in $S_i$ to some vertex in $T_i$. We therefore only need to bound the congestion.

Let $\beta_e$ be the probability that edge $e$ is used by at least $c + 1$ paths in $P'$. We have that $\Pr[\beta_e] \leq (\frac{1}{mg})^{c+1}$. We will apply the Lovasz Local Lemma, so we need to bound the number of events $\beta_{e'}$ that are dependent on $\beta_e$. These two events are dependent if and only if $e$ is on path $P$, $e'$ is on path $P'$, and $P, P'$ belong to the same subset in the partition. Since edge $e$ is contained in at most $m$ paths and each subset of the partition contains $mg$ paths, the number of dependent events is at most $m(mg)n = m^2 gn$. By choosing $g = \tilde{O}(n^{1/c})$, we can guarantee that the LLL condition holds.
5.3 Relationship between Problems

There are many problems that seem to be related to our integral concurrent flow problem. In this thesis, we have shown that basic-ICF generalizes Congestion Minimization, and since we can adapt the ideas from [41] to prove the hardness of group-ICF, this means that group-ICF may be closely related to Machine Minimization in a way that is currently unclear to us. Even when \( D > k \), group-ICF still captures the hardness of EDP problem, so our polylogarithmic approximation result for this case is essentially tight (within a poly-logarithmic factor).

\[ \text{Max-Min Interval Scheduling} \rightarrow \text{Machine Minimization} \rightarrow \text{group-ICF (when} D > k \text{)} \rightarrow \text{EDP} \rightarrow \text{Congestion Minimization} \]

**Figure 5.2:** The connections between problems related to ICF. The arrows indicate that one problem is a special case of another, and the line with ? indicates an unknown (but seemingly existing) relationship.

5.4 Conclusion and Open Problems

We introduce a new graph splitting theorem that allows us to design approximation algorithms for both basic-ICF and group-ICF. This theorem allows us to invoke the EDP algorithm iteratively in the graph without accumulating congestion. We believe that the
1. We have shown that Congestion Minimization is a special case of basic-ICF, but the connection between these two problems are not yet understood. It is interesting to see if there is a fundamental difference between the two problems. For instance, Congestion Minimization has $O\left(\frac{\log n}{\log \log n}\right)$ approximation algorithm and $\Omega(\log \log n)$ lower bound. Can we prove a stronger lower bound for basic-ICF? Even a stronger lower bound on the intregrality gap would be interesting.

2. Can we get an $o(n^{1/2-\epsilon})$ approximation algorithm for basic-ICF with no congestion. Unlike EDP and group-ICF, basic-ICF does not suffer from $\Omega(\sqrt{n})$ integrality gap lower bound. Even poly log $n$ approximation algorithm might be possible.

3. The proof of our graph splitting theorem uses the EDP algorithm by Chuzhoy [40] as a sub-routine. It is interesting to find a direct proof that does not rely on this theorem.

4. We show that the group-ICF problem is very hard to approximate when $D << k$, and when $D >> k$, we are able obtain poly log $n$ approximation algorithm with constant congestion. What happens when $k$ is a small constant? Can we obtain a poly-logarithmic approximation algorithm (with no congestion) for group-ICF when, says, $k = 2$? We note that resolving this question requires a novel idea since the integrality gap is $\Omega(\sqrt{n})$ even for $k = 2$.

5. We have shown that group-ICF generalizes EDP, but we currently do not know connection between basic-ICF and EDP. Our results show that, with a non-trivial amount of work, we can use an EDP algorithm by Chuzhoy to get an algorithm for basic-ICF. Can we get an approximation algorithm for basic-ICF without using EDP algorithms? Or, conversely, can we get an approximation algorithm for EDP assuming that we have an algorithm for basic-ICF?
Part II

Maximum Independent Set of Rectangles
CHAPTER 6
PRELIMINARIES

6.1 Preliminaries

In our setting, a rectangle $R$ is specified as a quadruple $(x_l(R), x_r(R), y_t(R), y_b(R))$ of real numbers, corresponding to the $x$-coordinates of its left and right boundaries and the $y$-coordinates of its top and bottom boundaries respectively. Furthermore, we assume that each rectangle is closed, i.e. each $R \in \mathcal{R}$ is defined as $R = \{(x, y) : x_l(R) \leq x \leq x_r(R) \text{ and } y_b(R) \leq y \leq y_t(R)\}$. We say that rectangles $R$ and $R'$ intersect iff $R \cap R' \neq \emptyset$.

Intersection types: We will distinguish between three types of intersections: corner, crossing, and containment (see Figure 6.1) whose formal definitions are as follows. For two overlapping rectangles $R, R'$, if $R$ does not contain any corner of $R'$, and $R'$ does not contain corner of $R$, then we say that the intersection is crossing. Otherwise, if $R' \subset R$ or $R \subset R'$, then it is a containment intersection. Otherwise, it is a corner intersection.

![Figure 6.1: Three possible intersection types](image)

113
Polynomially Bounded Weights: We argue that we can assume, by losing a constant factor in the approximation ratio, that all weights $w_R$ are positive integers of values at most $2n$. We first scale the weights of rectangles so that the minimum weight is at least 1. Let $W_{\text{max}}$ be the weight of the maximum-weight rectangle. For each rectangle $R \in \mathcal{R}$, we assign a new weight

$$w'_R = \left\lfloor w_R \cdot \frac{2n}{W_{\text{max}}} \right\rfloor$$

In the new instance, the weight of maximum weight rectangle becomes $2n$. It is easy to see that any $\gamma$-approximate solution to the new instance gives a $2\gamma$-approximate solution to the original instance.

Graph theoretic notions: Since we have an intersection graph that represents the problem instance $\mathcal{R}$ of MISR, we can talk about many graph theoretic concepts, such as clique, coloring, and degeneracy of graphs, in this geometric setting.

First, a set of rectangles $Q \subseteq \mathcal{R}$ forms a clique if the intersection of all the rectangles in $Q$ is non-empty. The maximum clique size of instance $\mathcal{R}$ is the maximum number of rectangles in $\mathcal{R}$ whose intersection is non-empty. In the context of rectangles, this notion of “geometric clique” corresponds to the standard notion of cliques in graph theory, since it is well known that rectangles satisfy Helly property, that is, any sub-collection of pairwise intersecting rectangles must contain a common point in the plane.

Let $\mathcal{R}'$ be a collection of rectangles. We say that $\mathcal{R}'$ admits a c-coloring (or is c-colorable) iff there exists an assignment $\lambda : \mathcal{R}' \rightarrow [c]$ such that for any $R, R' \in \mathcal{R}'$ if $R \cap R' \neq \emptyset$, then $\lambda(R) \neq \lambda(R')$. A collection $\mathcal{R}'$ of rectangles is k-degenerate if for every sub-collection $\mathcal{R}'' \subseteq \mathcal{R}'$, there exists a rectangle $R \in \mathcal{R}''$ such that $R$ intersects with at most $k$ other rectangles in $\mathcal{R}''$. It is a standard fact that any k-degenerate collection $\mathcal{R}'$ is $(k+1)$-colorable, and such coloring can be computed efficiently: Choose a rectangle $R \in \mathcal{R}'$ such that the size of neighbors of $R$, i.e. $|\{S \in \mathcal{R}' : S \cap R \neq \emptyset\}|$, is at most $k$. Recursively color the collection $\mathcal{R}' \setminus \{R\}$. Assign any color to $R$ that does not conflict with any of $R$’s neighbors.
6.2 Maximum Independent Set via Coloring

We solve the MISR problem through its connection with another problem, that we call *Rectangle Coloring* (RCOL). In this problem, we are given a collection of rectangles $\mathcal{R}$ in the plane, and the goal is to compute the minimum number $c$, together with a corresponding coloring $\lambda : \mathcal{R} \rightarrow [c]$, such that $\mathcal{R}$ is $c$-colorable.

In this section, we discuss the connections between MISR and RCOL. We remark that any $c$-coloring algorithm for $\mathcal{R}$ trivially implies an algorithm that finds an independent set of size $|\mathcal{R}|/c$. However, this bound is too loose to say anything meaningful; for example, if all rectangles intersect at the same point in the plane, this bound only allows us to get $n$-approximation. We instead focus on the connection via a natural LP relaxation of MISR, as given below.

\[(\text{MISR-LP}) \quad \max \sum_{R \in \mathcal{R}} w_R x_R \quad \text{s.t.} \quad \sum_{R : p \in R} x_R \leq 1 \text{ for all } p \in \mathbb{R}^2\]

We have, for each rectangle $R$, variable $x_R$ that indicates whether rectangle $R$ is selected in the solution. The constraint says that for each point in the plane, we select at most one rectangle containing such point. Note that we do not need to write these constraints for all points in the plane. We only need to consider only a polynomial number of “interesting points” as follows. Let $\mathcal{P}_x$ be the set of $x$-coordinates that correspond to the left or right boundary of some rectangle in $\mathcal{R}$, i.e. $\mathcal{P}_x = \{x^l(R), x^r(R) : R \in \mathcal{R}\}$. Similarly, we define $\mathcal{P}_y = \{y^t(R), y^b(R) : R \in \mathcal{R}\}$ to be the set of $y$-coordinates that correspond to the top or bottom boundary of some rectangle in $\mathcal{R}$, and let $\mathcal{P} = \mathcal{P}_x \times \mathcal{P}_y$. Clearly $|\mathcal{P}| \leq (2n)^2$. It is sufficient for us to write the constraints only for these points. Therefore, we have the following (polynomial-sized) LP relaxation.
\[(\text{MISR-LP}) \quad \max \sum_{R \in \mathcal{R}} w_R x_R \]
\[\text{s.t. } \sum_{R : p \in R} x_R \leq 1 \text{ for all } p \in \mathcal{P} \]

Now we are ready to precisely state the connection between MISR and RCOL, through LP relaxation. For any instance \(\mathcal{R}\), we denote by \(q(\mathcal{R})\), the maximum clique size of \(\mathcal{R}\), and \(\chi(\mathcal{R})\) the minimum number \(c\) such that \(\mathcal{R}\) is \(c\)-colorable. It is well known that \(\chi(\mathcal{R}) \geq q(\mathcal{R})\).

We will now be interested in the extent to which these two quantities may differ.

Let \(\mathcal{G}\) be any family of rectangle instances. We define the function \(\sigma(\mathcal{G}, q)\) as

\[\sigma(\mathcal{G}, q) = \sup_{\mathcal{R} \in \mathcal{G} : q(\mathcal{R}) = q} \frac{\chi(\mathcal{R})}{q(\mathcal{R})}\]

For a fixed family \(\mathcal{G}\), the function \(\sigma(\mathcal{G}, q)\) is a function of the clique size \(q\). The following theorem shows that this quantity is closely related to the integrality gap of \((\text{MISR-LP})\).

**Theorem 32** Let \(\mathcal{G}\) be any family of instances. Then there is a constant \(c_1\) that does not depend on the input size such that for any \(\mathcal{R} \in \mathcal{G}\), we have that the integrality gap of \((\text{MISR-LP})\) on input \(\mathcal{R}\) is at most \(\sigma(\mathcal{G}, c_1 \log n)\). Moreover, if there is a polynomial time coloring algorithm giving such bound, then we have an \(\sigma(\mathcal{G}, c_1 \log n)\) randomized approximation algorithm for MISR.

**Proof:** For convenience, for any subset \(\mathcal{R}' \subseteq \mathcal{R}\) and LP-solution \(z\), we define the term volume, denoted by \(\text{vol}_z(\mathcal{R}') = \sum_{R \in \mathcal{R}'} w_R z_R\).

Let \(z\) be an optimal LP solution with associated volume \(\text{vol}_z(\mathcal{R}) = \text{OPT}\). Observe that if \(\text{OPT} \leq O(n)\), getting a constant integrality gap bound is trivial: simply output the maximum-weight rectangle, whose weight is always \(2n\). Therefore, we assume that \(\text{OPT} \geq 32n\). Let \(M = 64 \log n\). The next lemma states that we can convert \(z\) into solution
\[ z' \text{ that is } \left( \frac{1}{M} \right) \text{-integral having roughly the same volume as } z \text{ with high probability.} \text{ The proof only uses standard randomized rounding techniques.} \]

**Lemma 17** There is an efficient randomized algorithm that, given an optimal LP-solution \( z \) of value \( \text{vol}_z(\mathcal{R}) = \text{OPT} \geq 32n \) for \( \mathcal{R} \), produces with high probability, a feasible solution \( z' \) for (LP) that is \( \left( \frac{1}{M} \right) \)-integral, and \( \text{vol}_{z'}(\mathcal{R}) = \Omega(\text{OPT}) \).

**Proof:** We create a multi-set \( \mathcal{R}^* \) of rectangles (i.e. each rectangle \( R \) may appear in \( \mathcal{R}^* \) more than once) as follows. For each rectangle \( R \in \mathcal{R} \), we first create a set \( \mathcal{R}_R^* \), and then we define \( \mathcal{R}^* = \bigcup_{R \in \mathcal{R}} \mathcal{R}_R^* \). We also define a feasible LP solution \( z^* \) for rectangles in \( \mathcal{R}^* \).

For each \( R \in \mathcal{R} \), let \( c_R = \lceil x_R M \rceil \) and define set \( \mathcal{R}_R^* \) to contain \( c_R - 1 \) copies of \( R \) with the first \( c_R - 1 \) copies having associated LP-value of \( 1/M \), while the last copy \( R' \) has LP-value \( z_{R'}^* = z_R - (c_R - 1)/M \). Clearly, for each \( R \in \mathcal{R} \) the sum of LP-values \( \sum_{R' \in \mathcal{R}_R^*} z_{R'}^* = z_R \). We now construct the multi-set \( \mathcal{R}' \subseteq \mathcal{R}^* \) where each (copy of) rectangle \( R' \in \mathcal{R}^* \) is independently selected to appear in \( \mathcal{R}' \) with probability \( \frac{M}{2} z_{R'}^* \). Finally, given the set \( \mathcal{R}' \), let \( z' \) be the LP-solution for collection \( \mathcal{R}' \), where each copy of rectangle in the set is assigned an LP-value of \( 1/M \).

For each rectangle \( R \) in the original instance \( \mathcal{R} \), we assign the new LP-value \( z'_R = \sum_{R' \in \mathcal{R}_R'} z'_{R'} \). It is clear that \( z' \) is \( (1/M) \)-integral. There are two claims we need to show: First we show that the solution \( z' \) is feasible with high probability, and next we show that the solution has volume \( \Omega(\text{OPT}) \) with high probability.

**Claim 10** The solution \( z' \) is feasible for \( \mathcal{R} \) with probability at least \( 1 - 1/n \).

**Proof:** Recall that the set \( \mathcal{P} \) is the set of interesting points in the plane. For any point \( p \in \mathcal{P} \), let \( \mathcal{C}_p \) denote the set of rectangles in \( \mathcal{R}' \) containing \( p \). It is enough to show that, for each point \( p \in \mathcal{P} \), the probability that \( |\mathcal{C}_p| > M \) is at most \( 1/n^4 \). First observe that \( \mathbb{E} [|\mathcal{C}_p|] \leq M/2 \). Using Chernoff bound, we get

\[
\Pr \left[ |\mathcal{C}_p| \geq M \right] \leq e^{-M/16} \leq 1/n^4
\]

117
Applying the union bound over the $n^2$ points in $P$, we get the desired result.

It remains to show that the value of the new solution is sufficiently large with high probability. That is, we will show that $\sum_{R \in \mathcal{R}'} w_{R} z_{R}' \geq \Omega(OPT)$ with high probability. We will need the fact that the rectangle with maximum weight has weight $2n$ and that $OPT \geq 32n$. Let $K = \lceil \log(2n) \rceil$. For the purpose of analysis, we partition rectangles in $\mathcal{R}^*$ into $K$ sub-collections $\{\mathcal{R}^*_i\}_{i=1}^K$ where $\mathcal{R}^*_i = \{R \in \mathcal{R}^* : w_R \in [2^{i-1}, 2^i)\}$. We also similarly partition the resulting set $\mathcal{R}'$ into $\{\mathcal{R}'_i\}_{i=1}^K$. Notice that we can associate each copy in $\mathcal{R}$ create a multi-subset $R \times K$.

Claim 11: $\sum_{i \in B} \sum_{R \in \mathcal{R}^*_i} w_{R} z_{R}' \leq OPT/2$.

Proof: Notice that for each bad index $i \in B$, $\sum_{R \in \mathcal{R}^*_i} w_{R} z_{R}^* < 2^i \left(\sum_{R \in \mathcal{R}^*_i} x_{R}\right) < 4 \cdot 2^i$. Summing over all $i \in B$, we get $\sum_{i \in B} 4 \cdot 2^i \leq \sum_{i=1}^{K} 4 \cdot 2^i \leq 16n \leq OPT/2$.

For good indices $i \notin B$, we have $E[|\mathcal{R}'_i|] = \frac{M}{4} \cdot \left(\sum_{R \in \mathcal{R}^*_i} z_{R}^*\right) \geq 2M$, so we can apply Chernoff bound to get

$$Pr \left[ |\mathcal{R}'_i| \leq \frac{M}{4} \cdot \left(\sum_{R \in \mathcal{R}^*_i} z_{R}^*\right) \right] \leq \frac{1}{n}$$

We can then apply the union bound to ensure that $|\mathcal{R}'_i| \geq \frac{M}{4} \cdot \left(\sum_{R \in \mathcal{R}^*_i} z_{R}^*\right)$ for all $i \notin B$ with high probability. So the sum of LP-values $z'$ in $\mathcal{R}'$, i.e. $\sum_{i \in \mathcal{R}'} z_{R}'$, is at least $|\mathcal{R}'|/M \geq \left(\sum_{R \in \mathcal{R}^*_i} z_{R}^*\right)/4$. For each $i \notin B$, we have $\sum_{R \in \mathcal{R}^*_i} w_{R} z_{R}' \geq 2^{i-1} \sum_{R \in \mathcal{R}^*_i} z_{R}^* \geq \frac{1}{8} \sum_{R \in \mathcal{R}^*_i} z_{R}^* w_{R}$. Summing over all $i \notin B$, we get the volume of $\text{vol}_{z'}(\mathcal{R}) \geq OPT/16$.

Now we use the lemma to finish the proof of Theorem 32. Given an LP solution $z'$, we create a multi-subset $\mathcal{R}'$ of $\mathcal{R}$ as follows: for each $R \in \mathcal{R}$, add $c_{R} = M z_{R}'$ copies of $R$ to $\mathcal{R}'$. Notice that we can associate each copy in $\mathcal{R}'$ with an LP-weight of $1/M$, so the maximum clique size is at most $M$. Moreover the volume in $\mathcal{R}'$ is $\text{vol}_{z'}(\mathcal{R}') = \sum_{R \in \mathcal{R}'} w_{R} z_{R}' = \Omega(OPT)$. Now assume that we have a $qf(q)$-coloring algorithm for any collection of rectangles with clique size $q$, where $f(q)$ is the function $\sigma(G, q)$. By invoking this algorithm on $\mathcal{R}'$, we divide
rectangles in $\mathcal{R}'$ into sets $\mathcal{R}'_1, \ldots, \mathcal{R}'_{Mf(M)}$ according to their colors. Let $\mathcal{R}'_j$ be the color class having maximum total volume among the sets $\{\mathcal{R}'_j\}$. We have that

$$\text{vol}_{z'}(\mathcal{R}'_j) = \sum_{R \in \mathcal{R}'_j} w_{Rz'} = \sum_{R \in \mathcal{R}'_j} \frac{w_R}{M} \geq \frac{\text{OPT}}{Mf(M)}$$

Therefore, the total weight of rectangles in $\mathcal{R}'_j$ is $\sum_{R \in \mathcal{R}'_j} w_R \geq \Omega(\text{OPT}/f(64 \log n))$, as desired. Notice that if we are satisfied with a non-constructive bound, we can simply invoke $f(M)M$ coloring of $\mathcal{R}'$ to get the integrality gap bound of $f(64 \log n)$.

\[\square\]

### 6.3 Our Results

In this thesis, we focus on obtaining approximation algorithms for unweighted MISR where all rectangles are assumed to have unit weight. In this case, we may assume that we do not have containment intersection: If some rectangle $R$ strictly contains $R'$, we may remove $R$ completely from the instance. Denote by $\mathcal{G}_{nc}$ a family of collections of rectangles in which there is no containment intersection.

Theorem 32 allows us to reduce MISR to RCOL, so we can focus on obtaining a coloring algorithm with a particular goal of bounding $\sigma(\mathcal{G}_{nc}, q)$. Our main result can be summarized in the following theorem.

**Theorem 33** Let $\mathcal{R}$ be a collection of rectangles with maximum clique size $q$, such that there is no containment intersection in $\mathcal{Q}$. Then $\mathcal{R}$ is $O(q \log q)$-colorable (or in other words, $\sigma(\mathcal{G}_{nc}, q) \leq O(\log q)$). Moreover, such coloring can be found in polynomial time.

Combining this with Theorem 32, we get the following result as a corollary.

**Theorem 34** There is a randomized $O(\log \log n)$ approximation algorithm for MISR when there is no containment intersection. In particular, there is an $O(\log \log n)$ approximation algorithm for unweighted MISR.
Using standard techniques, Theorem 34 can be extended to yield the following theorem.

**Theorem 35** There is a randomized $O(\log^{d-2} \log \log n)$ approximation algorithm for unweighted $d$-MISR.

We note that, after our $O(\log \log n)$ approximation algorithm for unweighted MISR was published, Chan and Har-peled obtained an $O(\log n/ \log \log n)$ approximation algorithm for the general case of MISR [32]. These bounds remain the best known.

### 6.4 Organization

We present an $O(q \log q)$ coloring algorithm in Chapter 7. It implies an $O(\log \log n)$ approximation algorithm for unweighted MISR. We show how to extend this result to higher dimensions in Section 8.1. Chapter 8 also contains applications of MISR in various areas of Computer Science, and the integrality gap example for (MISR-LP). We conclude this part of the thesis with a list of open problems.
CHAPTER 7
COLORING ALGORITHMS

In this section, we motivate the ideas used in the proof of our main result. We start by defining the notion of sparse instances which allow $O(q)$-coloring. However, some collections of rectangles are not sparse, and we need to deal with those instances differently. These two ideas together serve as basic building blocks of our algorithm. We then prove a weak bound, showing that any collection of rectangles is $O(q^2)$-colorable. Our main result can be viewed as a (non-trivial) extension of this algorithm.

7.1 Sparse Instances

We say that a collection of rectangles $\mathcal{R}$ is $s$-sparse if, for any rectangle $R \in \mathcal{R}$, there exist $s$ points $p_{1R}, p_{2R}, \ldots, p_{sR} \in P$ associated with rectangle $R$ (to be called representative points of $R$) such that for any overlapping rectangles $R, R' \in \mathcal{R}$, either $p_{iR}' \in R$ for some $i$ or $p_{jR} \in R'$ for some $j$. We note that Chan [30] uses similar ideas to define $\beta$-fat objects.

The following lemma can be seen as a generalization of Lemma XXX in [72]. The proof follows along similar lines.

**Lemma 18** Let $\mathcal{R}'$ be an $s$-sparse instance with maximum clique size $q$. Then $\mathcal{R}'$ is $2s(q-1)$-degenerate, and therefore is $(2s(q - 1) + 1)$-colorable.

**Proof:** Let $\mathcal{R}'$ be an $s$-sparse instance and $\mathcal{R}''$ be any sub-collection of $\mathcal{R}'$. It is enough to show that there exists a rectangle $R \in \mathcal{R}''$ such that $R$ overlaps with at most $2s(q - 1)$ other rectangles in $\mathcal{R}''$. Let $G[\mathcal{R}''] = (\mathcal{R}'', E)$ be an intersection graph defined on the sub-instance $\mathcal{R}''$, so it is sufficient to show that the number of edges in $E$ is at most $s(q - 1)|\mathcal{R}''|$. It will be convenient to think of edges in $E$ as directed edges where each edge $(R, R') \in E$ is directed from $R$ to $R'$ if $R$ contains some representative point of $R'$, and otherwise the edge is directed from $R'$ to $R$. Now consider any rectangle $R \in \mathcal{R}''$ and all incoming edges of $R$. 

121
Each incoming edge \((R', R)\) represents the fact that rectangle \(R'\) contains some representative point of \(R\). Since each representative point \(p_i^R\) can be contained in at most \(q - 1\) rectangles other than \(R\) itself, the number of incoming edges into \(R\) is at most \((q - 1)s\). This implies that the total number of edges is at most \(s(q - 1)|\mathcal{R}'|\).

We note that sparse instances are closely related to the notion of union complexity used extensively in computational geometry. Sparse instances capture a broad class of instances of intersecting geometric objects and set systems. We discuss some of them below.

**Squares:** Let \(\mathcal{R}_1\) be a collection of squares. For each rectangle \(R\), we define the four representative points of \(R\) to be its four corners. Notice that it is impossible for two squares to cross each other, so for any intersection between two squares, it must be the case that one square contains some corner of the other.

**Rectangles with bounded aspect ratios:** Let \(\mathcal{R}_2\) be a collection of rectangles in which each rectangle has the width of value between 1 and \(W\), where \(W\) is some constant. We claim that this collection of rectangles is \(O(W)\)-sparse, and therefore the collection \(\mathcal{R}_2\) is \(O(qW)\) colorable. This also implies an \(O(W)\) approximation algorithm for \(\text{MISR}\) in this case.

We define the representative points \(X_R\) for each rectangle \(R \in \mathcal{R}_2\) as follows. For each rectangle \(R\), let \(Q\) be the maximum-cardinality independent set among the rectangles \(R'\) that cross \(R\) and whose width is smaller than the width of \(R\). Notice that each such rectangle \(R'\) intersects \(R\) at four points, and we add these points to set \(X_R\). Finally we add the four corners of \(R\) to set \(X_R\).

### 7.2 Instances that are not sparse

A collection of rectangles, unfortunately, does not necessarily form a sparse instance, as shown in Figure 7.1 where we have \((n/2) \times (n/2)\) grid. Notice that the clique size of the instance is 2, while the degree of every rectangle is \(n/2\). Therefore this instance cannot
be $s$-sparse for any $s = o(n)$. An interesting point is that this instance only has crossing intersections, so this motivates us to investigate the complexity of the problem when the intersection types are limited to only crossing.

In the rest of this section, we study the colorability of rectangles when the intersection types are limited. It is clear that any collection of rectangles $\mathcal{R}$ in which two intersecting rectangles do not cross, forms a 4-sparse instance, and therefore is $O(q)$ colorable: Define the representative points of each rectangle $R$ to be its four corners. Therefore, if we do not allow crossing of rectangles, it is easy to do the coloring.

What if we only allow crossing but not any other types of intersections? We claim that this case is also easy to deal with.

**Lemma 19** Let $\mathcal{R}$ be a collection of rectangles in which crossing is not allowed. Then $\mathcal{R}$ is $q$-colorable.

**Proof:** We first notice that, for each point $p \in \mathcal{P}$, the collection of rectangles that contain $p$ can be ordered by their widths in increasing order (in this setting it is impossible for two intersecting rectangles containing $p$ to have the same width.)

Now we explain the coloring algorithm. Recall that $q$ is the maximum clique size of the instance $\mathcal{R}$. Our algorithm proceed in iterations. In the first iteration, let $Q \subseteq \mathcal{P}$ be the set
of “maximum clique points” in $P$, i.e. the points that are contained in exactly $q$ rectangles. We define the collection $R'$ to contain all rectangles whose widths are minimum for some point $p \in Q$. Observe that rectangles in $R'$ are disjoint and that $R \setminus R'$ does not have any clique of size $q$. So we color $R'$ using the same color, remove $R'$ from $R$, and proceed to the next iteration.

Each iteration guarantees that the size of maximum clique of the instance decreases by at least one. Therefore, after $q$ iterations we will be done, and the number of colors we use is exactly $q$.

Now we have two coloring algorithms that are used to deal with two different situations: (i) The algorithm for sparse instances, which is used when the rectangles do not have crossing, and (ii) The algorithm in Lemma 19, which can be used when only crossing intersection is allowed.

### 7.3 $O(q^2)$-Coloring Algorithm

Let $R$ be a collection of rectangles with maximum clique size $q$. We first show how to color $R$ using $O(q^2)$ colors. This coloring algorithm will be used later as a subroutine of our main result.

We define the function $v(R)$, the notion that quantifies the idea of “width” used in the previous section. For each rectangle $R \in R$, we denote by $V(R)$ the set of all rectangles $R' \in R$ such that $R$ and $R'$ cross each other, and the width of $R'$ is smaller than the width of $R$. Let $v(R)$ be the size of the maximum clique formed by the rectangles in $V(R)$. Notice that since the maximum clique size of $R$ is $q$, we have that $0 \leq v(R) \leq q - 1$ for all rectangle $R \in R$.

**Claim 12** For any rectangles $R, R' \in R$, if $v(R) = v(R')$, then $R$ cannot cross $R'$.

**Proof:** Assume for contradiction that $R$ crosses $R'$, and the width of $R$ is smaller than the width of $R'$. Let $Q \subseteq V(R)$ be a clique such that $|Q| = v(R)$. Then $\{R\} \cup Q \subseteq V(R')$ is a
clique, and therefore \( v(R') \geq v(R) + 1 \).

Using this claim, we are able to prove the following theorem.

**Theorem 36** Any collection of rectangles with maximum clique size \( q \) is \( O(q^2) \)-colorable.

**Proof:** Let \( \mathcal{R} \) be any collection of rectangles. We compute the values \( v(R) \) for all rectangles \( R \in \mathcal{R} \). Partition \( \mathcal{R} \) into \( q \) subsets \( S_1, \ldots, S_q \) where \( R \in S_i \) iff \( v(R) = i - 1 \). Since each set \( S_i \) does not contain any pair of crossing rectangles, it is 4-sparse, and so by Lemma 18 each such collection is \( O(q) \)-colorable. This implies that the set \( \mathcal{R} \) is \( O(q^2) \)-colorable.

The coloring result in Theorem 36 uses the values \( v(R) \) to define a “grouping” of rectangles into \( \{ S_i \}_{i=1}^q \) such that the intersection patterns of rectangles in the same set \( S_i \) are limited to only crossing and containment, which form a 4-sparse instance. Now we try to push this idea further. We would like to say such things as “if \( v(R) \) and \( v(R') \) are close, the intersection patterns of \( R \) and \( R' \) are limited”, and we would expect that if we group rectangles with roughly the same values of \( v(R) \) together, such collection should be “almost” sparse.

### 7.4 Structure Lemma

In this section, we introduce the important definitions and the key lemma. We start with the following lemma about the combinatorial structures of sets of intersecting rectangles.

**Lemma 20 (Structure Lemma)** Let \( \mathcal{C} \) be a clique, and \( R \) be any rectangle such that \( \mathcal{C} \subseteq V(R) \). Then we have that

\[
v(R) \geq \min_{R' \in \mathcal{C}} \{ v(R') \} + \lfloor |\mathcal{C}|/2 \rfloor
\]

In other words, for any sub-collection of rectangles \( \mathcal{R}' \) such that \( |v(R) - v(R')| \leq \delta \) for all \( R, R' \in \mathcal{R}' \), any clique \( \mathcal{C} \subseteq \mathcal{R}' \) of size larger than \( 2\delta \) is not a subset of \( V(R) \).

**Proof:** Let \( p = (x, y) \) be any point contained in the intersection of rectangles in \( \mathcal{C} \cup \{ R \} \). Consider now vertical line \( L \) passing through \( p \) (as shown in Figure 7.2). Let \( \mathcal{Q} \subseteq \mathcal{C} \) be
the set of \(\lfloor |C|/2 \rfloor\) rectangles whose left boundary is closest to \(L\) in \(C\), and let \(P \in Q\) be the rectangle whose right boundary is closest to \(L\) among the rectangles in \(Q\). Notice that all rectangles in \(C \setminus Q\) intersect the left boundary of \(P\), and all rectangles in \(Q \setminus \{P\}\) intersect the right boundary of \(P\). Let \(C'\) be a clique of size \(v(P)\) in \(V(P)\). This is the clique whose rectangles contribute to the value \(v(P)\). Observe that each rectangle in \(C'\) belongs to \(V(R)\), and that \(C\) is disjoint with \(C'\) since rectangles in \(C\) intersect the left or the right boundary of \(P\) while rectangles in \(C'\) do not. Let \(p' = (x', y)\) be any point in the intersection of rectangles in \(C' \cup \{P\}\) (the intersection region is shown as a black stripe in Figure 7.2) that is horizontally aligned with point \(p\). Assume first that \(x' > x\). Then every rectangle in \(Q\) contains \(p'\) because each rectangle in \(Q\) contains \(p\) (so its left boundary must lie on the left side of \(p\)) and intersects the right boundary of \(P\) (so its right boundary must be on the right of \(p'\)). Therefore \(C' \cup Q \subseteq V(R)\) form a clique of size at least \(v(P) + \lfloor |C|/2 \rfloor\). Similarly, if \(x' \leq x\), then every rectangle in \(C \setminus Q\) contains \(p'\), and we have that the set \(C' \cup (C \setminus Q) \cup \{P\}\) forms a clique of size at least \(v(P) + \lfloor |C|/2 \rfloor\).

![Figure 7.2](image)

Figure 7.2: Proof of Lemma 7.2 when \(x' > x\). The black stripe shows the intersection region of \(C'\)

We introduce the key definition of well-covered rectangles. Let \(R'\) be a sub-collection of rectangles. We say that rectangle \(R\) is \(\alpha\) vertically well-covered by \(R'\) if there exist two (not necessarily disjoint) collections of rectangles \(X_1, X_2 \subseteq R', |X_1| = |X_2| = \alpha\) such that (i) Each rectangle in \(X_1\) intersects the top boundary of \(R\), (ii) Each rectangle in \(X_2\) intersects
the bottom boundary of \( R \), and \( X_1 \cup X_2 \cup \{ R \} \) forms a clique. We denote by \( \alpha_{R'}(R) \) the maximum integer \( \alpha \) such that \( R \) is \( \alpha \)-well covered by \( R' \). When the choice of \( R' \) is clear from context, we write \( \alpha(R) \) instead of \( \alpha_{R'}(R) \).

We will often call the sets \( X_1 \) and \( X_2 \) the top and bottom coverage of \( R \) respectively. It is easy to see that \( \alpha_{R'}(R) \) can be computed in polynomial time: For each point \( p \in R \cap \mathcal{P} \), we compute the set of rectangles containing \( p \) and intersecting the top boundary of \( R \). Denote this set by \( X_p^1 \). Set \( X_p^2 \) is defined and computed similarly. Then we have that 
\[
\alpha_{R'}(R) = \max_{p \in R \cap \mathcal{P}} \min \{|X_p^1|, |X_p^2|\}.
\]

The following claim illustrates a connection between the notion of vertically well-covered rectangles and the maximum clique size of the instance.

**Claim 13** For any collection of rectangles \( R' \) that contains a clique of size \( q' \), there is at least one rectangle \( R \in R' \) such that \( \alpha_{R'}(R) \geq q'/2 - 1 \).

**Proof:** It is sufficient to find a rectangle \( R \) together with its top and bottom coverages \( X_1, X_2 \). Let \( C \subseteq R' \) be the clique of size \( q' \), and let \( X_1 \subseteq C \) denote the set of \( q'/2 - 1 \) rectangles with highest top boundaries (breaking ties arbitrarily). Then define \( X_2 \subseteq C \) as the set of \( q'/2 - 1 \) rectangles with lowest bottom boundaries in \( C \setminus X_1 \) (breaking ties arbitrarily). Let \( R \) be any rectangle in \( C \setminus (X_1 \cup X_2) \). It is easy to see that every rectangle in \( X_1 \) (resp. \( X_2 \)) intersects the upper (resp. lower) boundary of \( R \).

This immediately gives the following corollary.

**Corollary 4** For any collection \( R' \) of rectangles, let \( R'' \subseteq R' \) be a set of rectangles \( R \) that are \( \nu \) vertically well-covered by \( R' \), i.e. \( \alpha_{R'}(R) \geq \nu \), for some \( \nu > 2 \). Then the maximum clique size in \( R' \setminus R'' \) is at most \( 3\nu \).

**Proof:** Let \( \bar{R} = R' \setminus R'' \) be the set of remaining rectangles. Suppose a large clique of size \( 3\nu \) remains in \( \bar{R} \). Then by Claim 13, we would have a rectangle \( R \in \bar{R} \) with \( \alpha_{\bar{R}}(R) \geq 3\nu/2 - 1 > \nu \). And so we have \( \alpha_{R'}(R) > \nu \), a contradiction. \( \square \)
7.5 An $O(q^{3/2})$ Coloring for Restricted Setting

To see an immediate application of our key lemma and definition, we describe an $O(q^{3/2})$-coloring algorithm below. For simplicity of presentation, let us for now restrict the intersection types and assume that we do not have an intersection of rectangles $R$ and $R'$ such that $R$ contains at least two corners of $R'$. So now there are only two types of intersection: (i) crossing and (ii) corner intersection where one rectangle contains exactly one corner of another.

Our algorithm has three steps.

1. Compute the values $v(R)$ for rectangles $R \in \mathcal{R}$ in the beginning. These values are used throughout the algorithm.

2. Partition the rectangles into $\sqrt{q}$ sets $\{S_i\}_{i=1}^{\sqrt{q}}$ where rectangle $R$ belongs to $S_i$ iff $(i - 1)\sqrt{q} \leq v(R) < i\sqrt{q}$. So $|v(R) - v(R')| \leq \sqrt{q}$ for all $R, R'$ in the same set $S_i$.

3. For each $i : 1 \leq i \leq \sqrt{q}$,
   
   (a) Define $T_i = \{R \in S_i : \alpha_i(R) \geq 10\sqrt{q}\}$, where $\alpha_i(R)$ is the maximum integer $\alpha'$ such that $R$ is $\alpha'$ vertically well-covered by $S_i$.

   (b) $S'_i \leftarrow S_i \setminus T_i$.

   (c) Color $T_i$ and $S'_i$ using $O(q)$ colors.

Assuming that step 3(c) can be implemented, it is clear that the total number of colors used is $O(q^{3/2})$. It is therefore sufficient to show that sets $T_i$ and $S'_i$ are $O(q)$-colorable. For each set $S'_i$, notice that the clique size in $S'_i$ is at most $O(\sqrt{q})$ after the removal of $T_i$ from $S_i$, due to Corollary 4. Using the $O(q(\sqrt{q})^2)$-coloring algorithm from the previous section, we can get $O(q)$-coloring for each set $S'_i$. The following claim shows that we can also color $T_i$.

Claim 14 Each set $T_i$ is 5-sparse. Therefore, it is $O(q)$-colorable.
Proof: Recall that each $R \in T_i$ has $(i - 1)\sqrt{q} \leq v(R) < i\sqrt{q}$. We need to define, for each $R \in T_i$, five representative points $p_1^R, \ldots, p_5^R$. Now we fix $R$. Define $p_1^R, \ldots, p_4^R$ to be the four corners of $R$. Since $R$ is $(10\sqrt{q})$ vertically well-covered, denote by $X_1, X_2 \subseteq S_i, |X_1| = |X_2| = 10\sqrt{q}$ the top and bottom covers of $R$. Note that these rectangles may not be in $T_i$. Define $p_5^R$ to be any point in the intersection of rectangles in $X_1 \cup X_2 \cup \{R\}$. Consider two intersecting rectangles $R$ and $R'$ in $T_i$. If it is a corner intersection, we would be done. Otherwise, it is a crossing, and assume that the width of $R'$ is larger than the width of $R$, i.e. $R \in V(R')$. We claim that $p_5^R \in R'$: If not, assume without loss of generality that $p_5^R$ is below the bottom boundary of $R'$. Consider the “top $\alpha$-coverage” $X_1$ of $R$. Recall that all rectangles in $X_1$ contain $p_5^R$ and intersect the top boundary of $R$. Therefore, the only possible layout is that $R'$ crosses every rectangle in $X_1$ (because of our initial assumption), or in other words, $X_1 \subseteq V(R')$. Applying Lemma 20, we have that $v(R') \geq (i - 1)\sqrt{q} + 2\sqrt{q} = (i + 1)\sqrt{q}$, which is impossible. Notice that the proof of this claim would fail if we do not restrict the intersection types of rectangles. We fix this in the next section.

7.6 An $O(q \log q)$ Coloring

In this section, we give an $O(q \log q)$ coloring for any instance $R$ without containment intersection. We need another notion of covering. Let $R'$ be a collection of rectangles. We say that rectangle $R$ is $\tau$ horizontally well-covered by $R'$ if there is a set $X \subseteq R', |X| = \tau$ such that each rectangle $R' \in X$ contains the two left-corners of $R$ or the two right corners of $R$. We denote by $\tau_{R'}(R)$, the maximum value $\tau$ such that $R$ is $\tau$ horizontally well-covered by $R'$. Observe that given a collection $R'$ of rectangles, $\tau_{R'}(R)$ can be computed in polynomial time.

We are now ready to describe our algorithm. Let $R_0 = R$ and $\beta$ be an integer whose value will be specified later. Our algorithm proceeds in iterations. In iteration $i$, the algorithm
takes as input a collection of rectangles $\mathcal{R}_{i-1}$, together with its partition $\left\{ S_{i-1}^j \right\}_{j=1}^{\beta 2^{i-1}}$ where $S_{i-1}^j = \left\{ R \in \mathcal{R}_{i-1} : (j-1) \frac{q}{\beta 2^{i-1}} \leq v(R) < j \frac{q}{\beta 2^{i-1}} \right\}$. Moreover, the maximum clique size in each set $S_{i-1}^j$ is at most $q/2^{i-1}$ for all $1 \leq j \leq \beta 2^{i-1}$. Iteration $i$ will output a collection $\mathcal{R}_i \subseteq \mathcal{R}_{i-1}$ and a partition of $\mathcal{R}_i$ into $\left\{ S_i^j \right\}_{j=1}^{\beta 2^i}$, and the clique size in each set $S_i^j$ is at most $q/2^i$. Moreover, the collection of rectangles removed by iteration $i$, i.e. $\mathcal{R}_{i-1} \setminus \mathcal{R}_i$, is $O(q)$-colorable.

Assuming that such iterations can be implemented in polynomial time, it is clear how to get an $O(q \log q)$-coloring algorithm: Starting from $\mathcal{R}_0 = \mathcal{R}$, we first compute the values $v(R)$. We run the algorithm for $h = O(\log q)$ iterations until $q/2^h \leq 10$. Let $\mathcal{R}_h$ be the output from iteration $h$. Since $\mathcal{R} = \mathcal{R}_0 \supseteq \mathcal{R}_1 \supseteq \ldots \supseteq \mathcal{R}_h$, and $\mathcal{R}_{i-1} \setminus \mathcal{R}_i$ is $O(q)$-colorable for all $i$, we have that $\mathcal{R} \setminus \mathcal{R}_h$ is $O(q \log q)$-colorable. Now consider the collection $\mathcal{R}_h$ and its partition $\left\{ S_h^j \right\}_{j=1}^{\beta 2^h}$. The maximum clique size in each set $S_h^j$ is bounded above by some constant, so it is $O(1)$-colorable by Theorem 36. Combining all sets $S_h^j$ together (there are at most $q$ sets), we have that $\mathcal{R}_h$ is $O(q)$-colorable.

The iteration description is presented below.

**Observation 3** At the end of iteration $i$, $q(S_i^j) \leq q/2^i$ for all $j$.

It is clear that the above algorithm uses at most $O(q)$ colors. We only need to show that, for all $j$, sets $A_i^j$ and $S_i^j$ are $O(1)$-sparse. Denote by $\tilde{S}_i^j$, the set $S_i^j$ before the execution of line (3) of iteration $i$, i.e. before the removal of $A_i^j$ and $T_i^j$. This set $\tilde{S}_i^j$ will only be used for the purpose of analysis. We prove the following two lemmas.

**Lemma 21** Set $A_i^j$ is 5-sparse.

**Proof:** For each $R \in A_i^j$, we define five representative points as follows: The four points $p^R_1, \ldots, p^R_4$ are the four corners. Let $X_1, X_2 \subseteq \tilde{S}_i^j$ be a set of rectangles in the $\alpha_i(R)$-coverage of $R$; note that these rectangles in $X_1, X_2$ may not belong to $A_i^j$. We define the fifth point
Description of Iteration $i$:

1. Let $R_i = R_{i-1}$

2. Partition $R_i$ into $\{S_j^i\}_{j=1}^{\beta 2^i}$ where

$$S_j^i = \left\{ R \in R_i : (j - 1) \frac{q}{\beta 2^i} \leq v(R) < j \frac{q}{\beta 2^i} \right\}$$

3. For each rectangle $R \in S_j^i$, denote by $\alpha_i(R)$ the maximum $\alpha$ and $\tau$ such that $R$ is $\alpha$ vertically well-covered by $R'$. Similarly, let $\tau_i(R)$ be the maximum $\tau$ such that $R$ is $\tau$ horizontally well-covered by $R'$.

4. For each $j = 1$ to $\beta 2^i$
   - Define $A_j^i = \left\{ R \in S_j^i : \alpha_i(R) \geq q/2^{i+3}$ and $\tau_i(R) \leq q/2^{i+4} \right\}$
   - Define $T_j^i = \left\{ R \in S_j^i : \alpha_i(R) \geq q/2^{i+3}$ and $\tau_i(R) > q/2^{i+4} \right\}$.
   - $S_j^i \leftarrow S_j^i \setminus (A_j^i \cup T_j^i)$.
   - Color $A_j^i$ and $T_j^i$ using $O(q/2^i)$ colors.

Figure 7.3: The description of iteration $i$ of the algorithm
to be any point in the intersection of rectangles in $X_1 \cup X_2$. We argue that this choice of representative points actually works.

Consider two intersecting rectangles $R$ and $R'$ in $A_i^j$. If it is a corner intersection, we would be done. Otherwise, assume that the intersection is crossing, and the width of $R'$ is larger than the width of $R$. Assume that $p^R_5 \not\in R'$ and that $p^R_5$ lies below the bottom boundary of $R'$ (the other case when $p^R_5$ lies above the top boundary of $R'$ is similar.) Consider now the top coverage $X_1 \subseteq \tilde{S}_i^j : |X_1| \geq q/2^{i+3}$ of $R$. Let $Q$ be any rectangle in $X_1$. Since rectangle $Q$ intersects the top boundary of $R$ while $R'$ does not, either (i) $Q \in V(R')$, or (ii) $Q$ intersects the two left corners of $R'$ or the two right corners of $R'$. Let $X'_1 = \{Q \in X_1 : Q \in V(R')\}$ (i.e. rectangles in case (i)). And $X''_1 = X_1 \setminus X'_1$ (these are the rectangles in cases (ii)). The rectangles in $X''_1 \subseteq \tilde{S}_i^j$ contain a common point $p$, so they were considered in the computation of $\tau_i(R')$. Since $\tau_i(R') \leq q/2^{i+4}$, we get that $|X''_1| \leq q/2^{i+4}$, so $|X'_1| = |X_1| - |X''_1| \geq q/2^{i+4}$. Invoking Lemma 20 gives the following contradiction for $\beta \geq 2^6$:

$$v(R') \geq (j - 1) \frac{q}{\beta^2} + |X'_1|/2 > j \frac{q}{\beta^2}$$

See Figure 7.4 for illustration.

---

**Figure 7.4:** Rectangles $Q \in X''$ and $Q' \in X'$. 
Lemma 22 Set $T^j_i$ is $O(1)$-sparse.

Proof: For each $R$, we look at the structures of rectangles in $T^j_i$ that overlap $R$ and define the representative points for $R$. Now fix rectangle $R$ and let $Q_R$ be a maximum independent set of rectangles in $V(R) \cap T^j_i$ which is “minimal” in the sense that there is no other rectangle $R' \not\in Q_R$ that crosses rectangle $R'' \in Q_R$ and $R'$ has smaller width than $R''$. See Figure 7.5 for illustration. We note further that our algorithm does not need to compute $Q_R$. This definition is only used for defining the representative points, and as we show below, this set is small.

Claim 15 $|Q_R| \leq 200$

We prove the claim later. First let us finish the proof of Lemma 22 by defining representative points for rectangles in $T^j_i$. For each rectangle $R \in T^j_i$, the representative points of $R$ include (i) the four corners of $R$ and (ii) for each $R' \in Q_R$, the four intersection points between the boundaries of $R'$ and the boundaries of $R$. We show that these points indeed serve as representative points: For any two intersecting rectangles $R, R' \in T^j_i$, if they are not crossing, we would be done. Suppose $R'$ crosses $R$ such that the width of $R'$ is smaller than the width of $R$. Then, due to maximality of $Q_R$, there exists $R'' \in Q_R$ that overlaps with $R'$. Now due to the choice of $Q_R$ being minimal, rectangle $R'$ must contain some representative points on the boundaries of $R''$.

Proof: (Of Claim 15) Assume for contradiction that $|Q_R| > 200$. Consider a fixed rectangle $R' \in Q_R$. Recall that we have $\tau_i(R') \geq q/2^{i+4}$. Let $X_{R'} \subseteq \tilde{S}^j_i : |X_{R'}| \geq q/2^{i+4}$ be the set of rectangles in the horizontal coverage of $R'$; recall that rectangles in $X_{R'}$ share a common point $p$. Let $X'_{R'} \subseteq X_{R'}$ be the collection of rectangles in $X_{R'}$ that neither intersect the two left corners, nor do they intersect the two right corners of $R$. Notice that all rectangles in $X'_{R'}$ belong to $V(R)$. Since $X'_{R'}$ forms a clique, applying Lemma 20, we have that the size of $X'_{R'}$ cannot be too large, i.e. $|X'_{R'}| \leq q/2^{i+5}$. Now we define $Y_{R'} = X_{R'} \setminus X'_{R'}$, so we have $|Y_{R'}| = |X_{R'}| - |X'_{R'}| \geq q/2^{i+5}$. This is the set of rectangles that intersect the corners of $R$. 133
Figure 7.5: The rectangles in $Q_R$. The four intersection points between rectangles in $Q_R$ and $R$ denote the representative points of $R$.

Let $Y'R' \subseteq Y_R'$ be the set of rectangles in $Y_R'$ that intersect the two left corners of $R$, while $Y''R''$ be those intersecting the two right corners of $R$. We have that $|Y'R'| + |Y''R'| \geq |Y_R'|$. We say that $R'$ “contributes to the left of $R$” if $|Y'R'| \geq |Y_R'|/2 \geq q/2^{i+6}$; otherwise, we have that $|Y''R'| \geq q/2^{i+6}$ and we say that $R'$ contributes to the right.

Let $Q'_R \subseteq Q_R$ be the set of rectangles $R'$ that contribute to the left. We assume that $|Q'R'| > 100$ (otherwise, we have a symmetric case when $Q''_R = Q_R \setminus Q'_R$ has $|Q''_R| > 100$.) Notice that for two distinct rectangles $R', R'' \in Q'_R$, we have $Y_R' \cap Y_R'' = \emptyset$. We define $Y' = \bigcup_{R' \in Q'_R} Y_R'$ and, since sets $Y_R'$ are disjoint, we must have $|Y'| > q/2^i$. This implies a contradiction because rectangles $R' \in Y'$ contain a common point (i.e. the left corners of $R$), and since $Y' \subseteq \bar{S}_i^j$, they cannot form such a large clique. This completes the proof of the claim. 

\[ \square \]
CHAPTER 8
EXTENSIONS AND CONCLUDING REMARKS ON MISR

8.1 MISR in Higher Dimension

In this section, we show how to extend our algorithm in the plane to handle the problem in higher dimensions. More specifically, we prove the following theorem, whose proof uses essentially the same argument as in [3].

**Theorem 37** For any \( d \geq 1 \), if there is an \( f(n) \) approximation algorithm for \( d \)-MISR, then there is an \( O(f(n) \log n) \) approximation algorithm for \( (d+1)\)-MISR.

**Proof:** Let \( \mathcal{R} \) be an instance of \( (d+1)\)-MISR, and \( A \) be an algorithm that gives \( f(n) \) approximation algorithm for \( d \)-MISR. For each rectangle \( R \in \mathcal{R} \), let \( \gamma_R \) be the largest value of \( (d+1) \)th coordinate of any point in \( R \). Let \( y \in \mathbb{R} \) be the real number such that exactly \( \lfloor |\mathcal{R}|/2 \rfloor \) rectangles \( R \) of \( \mathcal{R} \) satisfy \( \gamma_R \leq y \). We partition \( \mathcal{R} \) into three sets, \( \mathcal{R}_<, \mathcal{R}_>, \mathcal{R}_= \) as follows. Let \( H \) be the hyperplane \( x_{d+1} = y \). Set \( \mathcal{R}_= \) contains all rectangles \( R \) that intersect \( H \), while set \( \mathcal{R}_< \) contains all rectangles \( R \) with \( \gamma_R < y \), and \( \mathcal{R}_> \) is defined as \( \mathcal{R} \setminus (\mathcal{R}_= \cup \mathcal{R}_<) \).

Let \( \mathcal{R}' \) be an instance of \( d \)-MISR obtained from \( \mathcal{R}_= \) by ignoring the \( (d+1) \)th coordinate of each rectangle, where \( \varphi: \mathcal{R}' \rightarrow \mathcal{R}_= \) is a one-to-one map between the two instances. It is easy to see that \( \mathcal{S} \subseteq \mathcal{R}' \) is an independent set if and only if \( \varphi(\mathcal{S}) \subseteq \mathcal{R}_= \) is an independent set. We run the algorithm \( A \) on the input \( \mathcal{R}' \) and solve the problem recursively on \( \mathcal{R}_< \) and \( \mathcal{R}_> \). Let \( \mathcal{T}, \mathcal{T}', \mathcal{T}'' \) be the corresponding solutions obtained from solving \( \mathcal{R}', \mathcal{R}_< \) and \( \mathcal{R}_> \), respectively. The algorithm outputs either \( \varphi(\mathcal{T}) \) or \( \mathcal{T}' \cup \mathcal{T}'' \), whichever contains more total weight of rectangles.

We show by induction on the input size that the algorithm produces an \( O(f(n) \log n) \) approximation algorithm. We consider two cases. First, if the optimal solution cost on \( \mathcal{R}' \) is at least \( \text{OPT} \log n \), then we have that \( w(\varphi(\mathcal{T})) \geq \text{OPT}/f(n) \log n \). Otherwise, denote by \( \text{OPT}' \) and \( \text{OPT}'' \) the total weight of the maximum-weight independent sets in \( \mathcal{R}_< \) and \( \mathcal{R}_> \).
respectively. We have that $\text{OPT}' + \text{OPT}'' \geq \text{OPT}(1 - \frac{1}{\log n}) = \text{OPT}\left(\frac{\log n - 1}{\log n}\right)$. Using the induction hypothesis, since $|R_<|, |R_>| \leq n/2$, we have that $w(T') \geq \text{OPT}'/f(n)(\log n - 1)$ and $w(T'') \geq \text{OPT}''/f(n)(\log n - 1)$. Therefore, $w(T' \cup T'') \geq \text{OPT}/f(n) \log n$.

\section{Applications and Connections to Other Problems}

As we mentioned in the beginning, MISR has connections to many problems. In this section, we discuss some of the applications and connections.

\textit{Unsplittable Flow on Line Graphs ([21])}

In the unsplittable flow problem on a path (UFP from now on), we are given a capacitated path $P$ and $n$ tasks, each of which having a demand, a profit and start and end vertices. The goal is to compute a maximum profit set of tasks, such that for each edge $e$ of $P$, the total demand of selected tasks that use $e$ does not exceed $c(e)$. This problem has received a lot of attentions in the past decade (see, e.g., [15, 16, 34, 21]), eventually leading to a constant approximation ratio [21].

Bonsma et. al. showed a very interesting connection between UFP and MISR. In particular, they show that UFP can be reduced to the following special case of MISR: We are additionally given a rectilinear curve $C$, where each rectangle is guaranteed to be drawn as high as possible with respect to $C$, i.e. the top boundary of each rectangle touches the curve $C$. Our goal is to solve MISR on this specific instance, called \textit{top-drawn instances}. Independently of [21], we managed to show an $O(\log \log n)$ approximation algorithm for top-drawn instances [25].

A constant approximation algorithm for MISR on top-drawn instances results in a constant approximation for MISR. More recently, it came to our attention [33] that an integrality gap upper bound of $\alpha$ for MISR would also imply an integrality gap of $O(\alpha)$ for LP relaxation.
used by [34].

**Map Labeling ([3, 46, 45])**

Suppose we have a map where we want to put labels of the cities on the map, and each label is represented by a rectangle. Due to a huge number of cities, it is sometimes inevitable to throw away some of the labels. Our goal is to place as many rectangular labels on the map as possible, without any conflicts. Agarwal, van Kreveld, and Suri [3] introduced this problem in 1997, argued that it can be solved by solving unweighted MISR, and proposed a number of algorithms to deal with it.

**Network Admission Control ([72, 1, 80])**

In admission control problem, we are given a network and a set \( R \) of \( n \) connection requests. Each request \( j \) is associated with a pair \((s, t)\) of source and destination in the network, time interval, profit, and a bandwidth requirement, or demand, \( d_j \in [0,1] \). Our goal is to select a subset of requests such that at any given time, the total demand on every link of network is at most one. When all demands are unit and the underlying network topology is a line graph, this problem corresponds exactly to MISR: The two axes correspond to the time and the edges on the line respectively. For each request \( j \), we have rectangle \( R_j \) whose projection on one axis corresponds to the time span of the request, while the other projection corresponds to the path of the request.

### 8.3 Integrality Gap of LP

Lewin-Eytan, Naor, and Orda shows that the integrality gap of natural LP relaxation for MISR is at most 4 when the instance does not have crossing. We show here an integrality gap of \( 3/2 \) for this special case.

In this section we show a lower bound on the integrality gap of (LP) that asymptotically
approaches 3/2. Our instance only contains corner intersections. Our construction is recursive, and we construct a sequence $I_1, \ldots, I_n$ of instances, where the integrality gap of $I_n$ is \[ \frac{3n+2}{2(n+1)}. \]

Each instance $I_j$ contains three special rectangles: $R_{1j}^j$, $R_{2j}^j$ and $R_{3j}^j$. Additionally, we have a “virtual” rectangle $R_{vj}^j$ that has the property that all rectangles of $I_j$ excluding $R_{1j}^j$, $R_{2j}^j$, $R_{3j}^j$ are contained in $R_{vj}^j$. We notice that $R_{vj}^j$ is not part of the problem instance, but is convenient to use as a bounding box for all rectangles in $I_j$ excluding $R_{1j}^j$, $R_{2j}^j$, $R_{3j}^j$. Figure 8.1 shows a schematic view of $I_j$.

![Figure 8.1: Schematic View of $I_j$](image)

Instance $I_1$ contains five rectangles, whose intersection graph is just a 5-cycle, as shown in Figure 8.2.

In order to obtain instance $I_{j+1}$ from instance $I_j$ we proceed as follows. First, we rotate $I_j$ clockwise by 90 degrees. Next, we place $R_{1j}^{j+1}$ to the left of $I_j$, intersecting $R_{1j}^j$ and $R_{2j}^j$ but not intersecting $R_{vj}^j$. Similarly, $R_{2j}^{j+1}$ is placed to the right of $I_j$, intersecting $R_{3j}^j$ but not $R_{vj}^j$. Finally, $R_{3j}^{j+1}$ is placed above $I_j$, intersecting $R_{1j}^{j+1}$ and $R_{2j}^{j+1}$ only. Figure 8.3 shows how instance $I_{j+1}$ is constructed from instance $I_j$, including the virtual rectangle $R_{vj}^{j+1}$.
We now proceed to analyze the integrality gap of instance $I_j$. First, it is clear that $I_j$ contains $3j + 2$ rectangles, and the size of the maximum clique is 2. Therefore, a fractional solution assigning LP-weight $\frac{1}{2}$ to each rectangle is a feasible solution of cost $(3j + 2)/2$. We next show that the cost of the integral solution is at most $j + 1$.

**Claim 16** The optimal integral solution of $I_j$ contains at most $j + 1$ rectangles.

**Proof:** We prove by induction on $j$. It is easy to verify that the optimal solution for $I_1$ contains 2 rectangles, and the optimal solution for $I_2$ contains 3 rectangles. Assume that maximum independent set for $I_j$ contains $(j + 1)$ rectangles and consider $I_{j+1}$. Let $S$ be
any solution for $I_{j+1}$. Notice that there are exactly three rectangles in $I_{j+1}$ that do not appear in $I_j$: $R_1^{j+1}, R_2^{j+1}$ and $R_3^{j+1}$. Therefore, if $S$ contains at most one of these three rectangles, we can use the induction hypothesis to conclude that $|S| \leq j + 2$. Assume now that $S$ contains at least two of these rectangles. The only way for this to happen is when both $R_1^{j+1}$ and $R_2^{j+1}$ are in $S$. Then none of the rectangles $R_3^{j+1}, R_1^j, R_2^j, R_3^j$ may belong to $S$. The size of $S \setminus \{R_1^{j+1}, R_2^{j+1}\}$ is then bounded by optimal integral solution for $I_{j-1}$, and therefore by the induction hypothesis, $|S| \leq j + 2$.

\[\square\]

It now follows that the integrality gap for $I_n$ is $(3n + 2)/2(n + 1)$.

### 8.4 Dealing with Containment Intersection

We know that, without crossing intersection, we have a $q$-coloring algorithm, and without containment, our main result gives $O(q \log q)$ coloring algorithm. Since our algorithm cannot deal with containment intersection, it is also an interesting question to ask whether we can get a good coloring algorithm for the case when we have containment and crossing but not corner intersection. This result implies that, if the instance does not have one of the three possible intersections, obtaining a coloring algorithm using sub-quadratic number of colors is possible.

**Theorem 38** Let $\mathcal{R}$ be a collection of rectangles with clique size $q$, in which the intersection types are only containment and crossing. Then $\chi(\mathcal{R}) = q(\mathcal{R})$. This implies that $\sigma(\mathcal{R}) = 1$ and maximum independent set of $\mathcal{R}$ can be found in polynomial time.

Our algorithm is simple: Sort the rectangles $R_1, \ldots, R_n$ by their top boundaries where $R_i$ has higher coordinate of top boundary than $R_j$ whenever $i < j$ (breaking ties arbitrarily). Then we iteratively color rectangles in this order. When $R_i$ is being considered, we assign the color of $R_i$ as the smallest positive number that does not create conflicts with any rectangle $R'$ overlapping with $R$ and preceding $R$ in the ordering. It is clear that this algorithm returns
a valid coloring. Below we argue that it uses no more than $q$ colors.

First we introduce another notion of coverage. For rectangle $R$, let $C(R)$ be the set of rectangles (completely) containing $R$ and $c(R)$ denote the size of set $C(R)$. Observe that $V(R) \cap C(R) = \emptyset$. It also follows from the definition that any clique formed by rectangles in $C(R) \cup V(R)$ has size at most $v(R) + c(R)$.

**Claim 17** For any rectangle $R$, $0 \leq v(R) + c(R) \leq q - 1$.

**Proof:** For any clique $C \subseteq V(R)$ such that $|C| = v(R)$, we have that the intersection of rectangles in $C$ has a common point with rectangles in $C(R) \cup \{R\}$. Since the maximum clique is at most $q$, the claim follows.

We are now ready to prove the theorem. It is sufficient to show that the number of colors used is at most $q$ which follows from the following lemma.

**Lemma 23** Let $\rho : \mathcal{R} \rightarrow \mathbb{N}$ be a coloring obtained from the above algorithm. Then, for any rectangle $R$ we have

$$\rho(R) \leq v(R) + c(R) + 1$$

In particular, $\rho(R) \leq q$.

**Proof:** For any rectangle $R$, we observe that rectangles in $V(R) \cup C(R)$ precede $R$ in the ordering. We will prove the lemma by contradiction. Let $R$ be the first rectangle (w.r.t. the ordering) for which $\rho(R) > v(R) + c(R) + 1$. By the time $R$ is being colored, all rectangles in $V(R) \cup C(R)$ have already been colored. The fact that $R$ is assigned color $\rho(R) > v(R) + c(R) + 1$ means that for all $i \in \{1, \ldots, v(R) + c(R) + 1\}$, there exists rectangle $R' \in V(R) \cup C(R)$ with $\rho(R') = i$. We will identify point $p \in R$ such that there are $v(R) + c(R) + 1$ rectangles with distinct colors in $V(R) \cup C(R)$ containing $p$; this is a contradiction because any clique in $V(R) \cup C(R)$ cannot be larger than $v(R) + c(R)$.

Let $Q_0 = R$. Given $Q_i$, we define $Q_{i+1}$ as the rectangle $R'$ in $V(Q_i)$ such that $\rho(R')$ is maximized. This allows us to define a sequence of rectangles $Q_1, \ldots, Q_l$ where $V(Q_l) = \emptyset$.
Let \( p \) be any point in \( \bigcap_i Q_i \). It suffices to show that for any \( i = 1, \ldots, c(R) + v(R) + 1 \), there exists a rectangle \( Q_j \) such that \( \rho(Q_j) = i \). First we prove a lemma that will be useful later.

**Lemma 24** For any \( i = 0, \ldots, l - 1 \), we have \( V(Q_{i+1}) \cup C(Q_{i+1}) \subseteq V(Q_i) \cup C(Q_i) \)

**Proof:** It is obvious that \( V(Q_{i+1}) \subseteq V(Q_i) \). Let \( R' \in C(Q_{i+1}) \), so \( R' \) is a rectangle that completely contains \( Q_{i+1} \). First it is clear that \( R' \) overlaps with \( Q_i \). Since we have only two types of intersections, rectangle \( R' \) either contains the whole \( Q_i \) (which implies that \( R' \in C(Q_i) \)) or \( R' \) intersects \( Q_i \) in non-corner manner and has smaller width (in which case \( R' \in V(Q_i) \). Thus, \( R' \in V(Q_i) \cup C(Q_i) \).

**Claim 18** For any \( 0 \leq i < l \), \( \rho(Q_i) > \rho(Q_{i+1}) \).

**Proof:** First, observe that we colored \( Q_{i+1} \) before \( Q_i \), and for all color class \( j \in [\rho(Q_{i+1}) - 1] \), we have some rectangle \( R' \in V(Q_{i+1}) \cup C(Q_{i+1}) \) such that \( \rho(R') = j \) (otherwise, the algorithm would have assigned a color class with lower number to \( Q_{i+1} \)). At the time the algorithm was trying to color \( Q_i \), rectangles in \( V(Q_{i+1}) \cup C(Q_{i+1}) \cup \{Q_{i+1}\} \) are among the potential conflicts (since they all intersect \( Q_i \)). Therefore, it is impossible for \( Q_i \) to use any color in \( [\rho(Q_{i+1})] \).

The above two claims imply that \( 0 \leq \rho(Q_i) < \ldots < \rho(Q_0) = \rho(R) \) and \( V(Q_i) \cup C(Q_i) \subseteq \ldots \subseteq V(Q_0) \cup C(Q_0) \). We complete the proof by showing the following claim.

**Claim 19** For all \( j \in [v(R) + c(R) + 1] \), there exists a rectangle \( R_j \in V(R) \cup C(R) \) such that \( \rho(R_j) = j \) and \( R_j \) contains \( p \).

**Proof:** If \( j = \rho(Q_i) \) for some \( i \), then we are done since rectangle \( Q_i \) contains \( p \) and belongs to \( V(R) \). Otherwise, since \( \rho(Q_0) > c(R) + v(R) + 1 \geq j \), we have two cases:

- **Case 1:** \( \rho(Q_{i+1}) < j < \rho(Q_i) \) for some integer \( i \). Since \( Q_i \) uses color \( \rho(Q_i) \) instead of a smaller index \( j \), there must be some rectangle \( R_j \in C(Q_i) \cup V(Q_i) \) such that \( \rho(R_j) = j \). This rectangle \( R_j \) cannot belong to \( V(Q_i) \) because \( Q_{i+1} \) is the rectangle

142
with highest color index among the rectangles in $V(Q_i)$. Hence $R_j \in C(Q_i)$, so $R_j$ completely contains $Q_i$, which contains point $p$.

• **Case 2:** $0 < j < \rho(Q_l)$. Since $V(Q_l) = \emptyset$, there must be a rectangle $R_j \in C(Q_l)$ such that $\rho(R_j) = j$. Rectangle $R_j$ contains $Q_l$ and the point $p$.

8.5 A Short Survey on Rectangle Coloring

We believe that the **Rectangle Coloring** problem is an interesting problem on its own (besides its implication to **MISR**), so we provide a short survey here. A more complete story can be found in an excellent survey by Kostochka [70].

Let $\mathcal{G}^*$ be a collection of rectangles. The study of the ratio between the chromatic number and clique number of rectangle intersection graphs, i.e. $\sigma(\mathcal{G}^*, q)$, began in 1948, when Bielecki [19] asked whether the value of $\sigma(\mathcal{G}^*, q)$ is independent of the instance size $n$. This question was answered positively by Asplund and Grünbaum in 1960 [13], when they show $\chi(\mathcal{R}) \leq 4q^2 - 3q$, which implies that $\sigma(\mathcal{G}^*, q) \leq 4q - 3$.

The bound was later improved to $\sigma(\mathcal{G}^*, q) \leq 3q - 2$ by Hendler [59], while the best lower bound remains $\sigma(\mathcal{G}^*, q) \geq 3$ by constructing a set of rectangles with clique size 2 but the chromatic number is 6 [13]; in fact, their result implies the exact bound $\sigma(\mathcal{G}^*, 2) = 6$. Better bounds are known for special cases. For squares, a better bound of $\sigma(squares, q) \leq 4$ was shown by Ahlswede and Karapetyan, and independently by Perepelitsa (see [4]). Lewin-Eytan et al. [72] and independently Agarwal and Mustafa [2] show that $\sigma(\mathcal{G}_{nc}, q) = 1$ where the collections of interest do not have any rectangle that contains corners of other rectangles (they both show that the corresponding intersection graphs are perfect graphs). All these bounds are constructive in the sense that they imply polynomial time algorithms for finding the colorings meeting such bounds. We refer to the survey by Kostochka for more detail on the related works in this area [70].
Finally, we note that the problem of finding an optimal coloring of rectangle intersection graphs is NP-hard [61].

8.6 Conclusion and Open problems

In this part of the thesis, we have discussed an \( O(\log \log n) \) approximation algorithm for unweighted MISR. This result was presented in SODA’09 paper by the author and Julia Chuzhoy [26]. The presentation in this thesis follows the rectangle coloring framework, as discussed in the followup paper [25]. In this framework, any \( qf(q) \) coloring algorithm can be transformed into an algorithm that gives \( f(\log n) \) approximation algorithm for MISR. Therefore, to get \( O(\log \log n) \) approximation algorithm for unweighted MISR, we only need to show \( O(q \log q) \) coloring algorithm for the instance with no containment intersection.

This paper follows the recent ideas in dealing with rectangle problems by using the “corner information” of the intersecting rectangles. We summarize the high-level ideas here. If the crossing intersections are not allowed, the resulting intersection graphs belong to what we call sparse instances, and therefore admit \( O(q) \) coloring. On the other hand, if we have only crossing, the resulting intersection graphs are perfect graph, so again we have \( q \)-coloring for the problem. Combining these two ideas immediately gives \( O(q^2) \) coloring algorithm. Our \( O(q \log q) \) coloring result can be seen as a clever way to combine the two ideas.

As of now, we are unable to extend our techniques to deal with general MISR, and this is because in the weighted setting, the containment intersection cannot be eliminated.

**Open problems:** We believe that both MISR and RCOL are interesting problems that have many connections to various other problems in computer science. The current best known approximation ratio for MISR is \( O(\frac{\log n}{\log \log n}) \) by Chan and Har-peled [32]. For unweighted MISR, the best known remains is our \( O(\log \log n) \) result [26]. For RCOL, the lower bound is \( \chi(\mathcal{R}) \geq 3q(\mathcal{R}) \), while the upper bound remains at \( \chi(\mathcal{R}) \leq O(q(\mathcal{R})^2) \). It is very interesting to see if \( \chi(\mathcal{R}) \) can be made sub-quadratic in terms of clique size. In this thesis, we show that
if there is no containment, \( O(q \log q) \) coloring is possible. It is not clear why containment makes the problem harder.

Improving any of the bounds listed above would be very challenging. We list below some of the specific directions that, we believe, are interesting.

1. One direction is to try to construct an integrality gap example. *Is there a super constant integrality gap example?* From Theorem 32, a super constant integrality gap implies a new coloring lower bound. Currently the best integrality gap lower bound we have is a factor of 3/2. There is still a very large gap between the upper and lower bounds.

2. Even special cases of MISR have been very well studied. Many special cases, such as squares, unit-height rectangles, fat rectangles, already admitted a PTAS. Another interesting special case is a top-drawn instance, as discussed in Section 8.2, which arises naturally in Unsplittable Flow problem on line graphs. *Is there an \( O(q) \) coloring algorithm for any top-drawn instance where \( q \) is the clique size of the input?* This will imply a constant integrality gap of an LP relaxation for Unsplittable Flow problem on line graphs.

3. In Section 8.1, we show a standard technique that allows us to extend any algorithm in \( d \)-dimensional problem to the \((d+1)\)-dimensional problem, losing a factor of \( \log n \) in the approximation factor. *Is there a better reduction?* More specifically, let \( g(n) \) be a function that is sub-logarithmic in \( n \). We are interested in obtaining the statement like “any \( \alpha \) approximation for \( d\)-MISR can be turned into \( \alpha g(n) \) approximation algorithm for \( (d+1)\)-MISR.

4. In Section 8.2, we discuss the connection between MISR and admission control in network. MISR corresponds to the admission control problem when the underlying network topology is a line graph. So our result immediately implies an improved approximation algorithm for admission control when all requests have the same profit. *Can our techniques be used to obtain an \( O(\log \log n) \) approximation algorithm for admission control*
problem on the trees? Currently the best known approximation ratio is $O(\log n)$, due to [72].
Part III

Resource Minimization for Fire Containment
CHAPTER 9  
PRELIMINARIES

9.1 Formal Problem Definition

In the RMFC problem, we are given an undirected graph \( G = (V, E) \), a source vertex \( s \in V \) where the fire starts, and a subset \( T \subseteq V \) of terminals that need to be protected. At each time step \( \tau \), we need to choose a subset \( U_\tau \) of at most \( k \) vertices to be saved. Once a vertex burns or is saved, it remains in this state permanently. Let \( B_\tau \) denote the set of burning vertices at time step \( \tau \), with \( B_0 = \{s\} \). Vertex \( v \) burns at time \( \tau + 1 \) iff it has not been saved so far, that is, \( v \not\in \bigcup_{1 \leq \tau' \leq \tau+1} U_{\tau'} \), and at least one neighbor of \( v \) is in \( B_\tau \). Our goal is to select subsets \( \{U_\tau\}_{\tau=1}^n \) of vertices to be saved at each time step \( \tau \), so that the fire does not spread to the terminals, while minimizing the maximum number of vertices to be saved at any time step, \( k = \max_\tau \{|U_\tau|\} \). Formally, given a solution \( S = \{U_\tau\}_{\tau=1}^n \), we say that a simple path \( P = (s, v_1, v_2, \ldots, v_z) \) is a fire spreading path w.r.t. \( S \) iff for each \( \tau : 1 \leq \tau \leq z \), \( v_\tau \not\in \bigcup_{\tau' \leq \tau} U_{\tau'} \). We say that vertex \( v \) burns in solution \( S \) iff it lies on some fire spreading path. Otherwise, we say that vertex \( v \) is protected by \( S \). A solution is feasible iff all terminals are protected.

9.2 LP Relaxation

For each vertex \( v \in V \), for each time step \( \tau \), we have a variable \( x_{\tau v} \) indicating whether \( v \) is saved at time step \( \tau \). In other words, \( x_{\tau v} \) is an indicator variable for \( v \in U_\tau \). For each terminal \( t \in T \), let \( P_t \) be the set of all paths connecting \( s \) to \( t \) in \( G \). We use the following LP relaxation.
\begin{align}
\text{(LP-1)} \quad \min \quad & k \\
\text{s.t.} \quad & \sum_{v \in V} x^\tau_v \leq k \quad \forall 1 \leq \tau \leq n \quad (9.1) \\
& \sum_{i=1}^z \sum_{\tau \leq i} x^\tau_{v_i} \geq 1 \quad \forall t \in T, \forall (s, v_1, \ldots, v_z = t) \in P_t \quad (9.2) \\
& x^\tau_v \geq 0 \quad \forall v \in V, \forall \tau \quad (9.3)
\end{align}

The first set of constraints bounds the number of vertices saved at each time step by $k$. The second set of constraints ensures that every terminal is protected, i.e. not on any fire-spreading path. Even though this LP has an exponential number of constraints, it can be solved in polynomial time, by using the following separation oracle. Clearly, constraints (9.1) and (9.3) can be checked in polynomial time. In order to produce a separation oracle for constraints of type (9.2), we build a dynamic programming table $T$. For each vertex $u \in V$, for each integer $1 \leq z \leq n$, there is an entry $T(u, z)$ in the table, whose value is the minimum, over all length-$z$ paths $P = (s, v_1, \ldots, v_z = u)$, of $\sum_{i=1}^z \sum_{\tau \leq i} x^\tau_{v_i}$. If no such path exists, the value of $T(u, z)$ is set to $\infty$. The table also stores the path $P$ that determined the value $T(u, z)$. The table is filled as follows. For $z = 1$, for each vertex $u \in V$ with $(s, u) \in E$, we set $T(u, 1) = x^1_u$. For all other vertices this value is set to $\infty$. For each vertex $u \in V$, value $T(u, z)$ for $z > 1$ is computed as follows: $T(u, z) = \min_{v: (v, u) \in E} \{ T(v, z-1) \} + \sum_{\tau=1}^z x^\tau_u$. If for some terminal $t \in T$, there is a value $1 \leq z \leq n$, for which $T(t, z) < 1$, then we know that the corresponding constraint is violated, and since the violating path is stored in the dynamic programming table, we can find it efficiently.

\section{9.3 Simplified LP for Special Cases}

In some graph classes, the above LP formulation can be simplified by incorporating the graph structure. In this thesis, we focus on two special graph classes: trees and directed layered graphs. We discuss a simplified LP for directed layered graphs below (recall that trees are
also directed layered graphs).

We assume that we are given a directed graph \( G = (V, E) \), whose vertices are partitioned into layers \( L_0, \ldots, L_\lambda \), where \( L_0 = \{s\} \), and all edges are between consecutive pairs of layers \( L_\tau, L_{\tau+1} \), directed from \( L_\tau \) towards \( L_{\tau+1} \). Let OPT denote the optimal solution cost. Notice that for any vertex \( v \in L_\tau \), all paths from \( s \) to \( v \) are of length \( \tau' \), and therefore, if \( v \) is not protected by \( S \), it must start burning at time \( \tau' \). There is no point to save \( v \) after time \( \tau' \), i.e. \( v \notin U_\tau \) for all \( \tau > \tau' \). We therefore have the following observation.

**Observation 4** Given any feasible solution \( \{U_\tau\}_{\tau=1}^n \) for RMFC on directed layered graphs, we can assume w.l.o.g. that for every \( \tau : 1 \leq \tau \leq \lambda \), \( U_\tau \subseteq \bigcup_{\tau' \geq \tau} L_{\tau'} \), and \( U_\tau = \emptyset \) for all \( \tau > \lambda \).

For vertex \( v \in V \), let \( P_v \) be the set of all paths connecting the source \( s \) to \( v \).

### Cuts and Amortized Cost

As suggested by Anshelevich et. al. [9], one can view this problem as “cuts over time”, in the following sense. Suppose \( S = \{U_\tau\}_{\tau=1}^n \) be a feasible solution for RMFC, and let \( U = \bigcup_{\tau=1}^n U_\tau \). It is easy to see that \( U \) is a feasible cut separating \( s \) from terminals \( T \): For any path \( P \) from \( s \) to some terminal \( t \in L_\tau \), we have that some vertex on \( P \) belongs to \( \bigcup_{\tau' \leq \tau} U_{\tau'} \subseteq U \).

Now we know that any solution in RMFC must be a feasible cut. The question is, if we have a feasible cut \( U \), can we transform \( U \) into a feasible solution \( S' = \{U_\tau\} \), and what would be the cost of solution \( S' \)? This question leads us to define the notion of “amortized cost” of a cut.

Let \( U \subseteq V \) be any feasible cut. The *amortized cost* of \( U \) is defined as the maximum, over all \( 1 \leq \tau \leq \lambda \), of \( \left( \sum_{\tau' \leq \tau} |U \cap L_{\tau'}| \right) / \tau \). The next claim shows that RMFC on directed layered graphs can be seen as selecting a feasible cut with minimum amortized cost.
**Claim 20** For any RMFC instance on directed layered graph, let $\text{OPT}$ denote the value of the optimum RMFC solution, and $\text{OPT}'$ the value of the cut with minimum amortized cost. Then, $\text{OPT} = \text{OPT}'$.

**Proof:** Let $\{U_\tau\}_{\tau=1}^\lambda$ be the optimal solution to the RMFC instance, and let $U = \bigcup_{\tau=1}^\lambda U_\tau$. From Observation 4, $\sum_{\tau' \leq \tau} |U \cap L_{\tau'}| \leq \sum_{\tau' \leq \tau} |U_{\tau'}| \leq \tau \text{OPT}$ for all $\tau$, so the amortized cost of $U$ is at most $\text{OPT}$.

We now show the converse direction. Let $U$ be a feasible set with amortized cost $k = \text{OPT}'$. We construct an RMFC solution $\{U'_\tau\}_{\tau=1}^\lambda$ with cost at most $k$ as follows. Consider first the solution $S = \{U_\tau\}_{\tau=1}^\lambda$, where $U_\tau = U \cap L_{\tau}$ for all $\tau$. It is easy to see that $S$ is a feasible solution: assume otherwise, and let $t$ be any terminal that is not protected by $S$. Let $P = (s, v_1, \ldots, v_z = t)$ be the corresponding fire spreading path. Since $U$ is a feasible set, there is some $v_\tau \in P \cap U$, for $1 \leq \tau \leq z$. Since $v_\tau \in U_\tau$, path $P$ cannot be a fire spreading path.

The cost of solution $S$ is $\max_\tau |U \cap L_{\tau}|$, which can be very large. We transform the solution $S$ to ensure that its cost is at most $k$, as follows. We process the sets $U_\lambda, U_{\lambda-1}, \ldots, U_1$ in this order. Let $U_\tau$ be the current set. If $|U_\tau| = k' > k$, then we remove any collection of $k' - k$ vertices from $U_\tau$ and add them to $U_{\tau-1}$. It is easy to see that this operation does not increase the amortized cost of $U$ and preserves the feasibility of the solution $S$. Once all the sets $U_\tau$ are processed, each of the resulting sets contains at most $k$ vertices. \hfill $\square$

**A Simplified LP for Directed Layered Graphs**

We can therefore focus on finding a feasible cut with a low amortized cost, which leads to the following LP relaxation.
The first constraint bounds the amortized cost, while the second constraint says that the chosen set of vertices forms a cut separating $s$ and vertices in $T$. Though the number of constraints in this LP is exponential, it can be efficiently solved using a separation oracle: We compute the node-weighted shortest path starting from $s$, where the weight of node $v$ is $x(v)$. If there is any path from $s$ to $t \in T$ of length less than one, then the cut constraint is violated.

**A Simplified LP for Trees**

When the underlying graph is a tree, an LP can be even further simplified as follows. First, for each terminal $t$, there is only one path from $s$ to $t$ in the tree. We can also simplify the constraint about amortized costs by using the following claim.

**Claim 21** Let $U$ be a feasible cut for tree $G$. The following two statements are equivalent:

1. The amortized cost of $U$ is at most $k$.
2. $|U \cap L_\tau| \leq k$ for all $1 \leq \tau \leq \lambda$.

**Proof:** One direction is clear: Any feasible cut $U$ such that $|U \cap L_\tau| \leq k$ for all $\tau$ immediately implies that the amortized cost is at most $k$.

Conversely, consider any feasible cut $U$ with amortized cost at most $k$. Whenever there is a layer $L_\tau$ such that $U \cap L_\tau$ contains more than $k$ vertices, we do the following. Let $L_\tau$...
be the layer with maximum \( \tau \) such that \(|U \cap L_{\tau}| = k' > k\). We modify the cut \( U \) as follows: Remove arbitrary \( k' - k \) vertices from \( U \cap L_{\tau} \) and add the ancestors of the removed vertices instead. Denote the new set by \( U' \). It is easy to see that the amortized cost of \( U' \) is at most \( k \), and \(|U' \cap L_{\tau'}| \leq k \) for all \( \tau' \geq \tau \). This allows us to make progress while maintaining the amortized cost of the cut.

In words, this claim says that the optimal solution always chooses at most \( k \) vertices from each layer if the input instance is a tree. We note that this is not true in directed layered graph, where the solution might want to choose the layer that contains small number of vertices (See Figure 9.1 where there is a solution of cost 1 that does not choose any vertex in the first layer).

Later in this thesis, we will talk about a feasible solution of cost \( k \) using these two notions interchangeably, i.e. a feasible solution is viewed as \( S = \{U_{\tau}\} \) where \( U_{\tau} \subseteq L_{\tau} \) for all \( \tau \), and a feasible cut \( U \) such that \(|U \cap L_{\tau}| \leq k \) for all \( \tau \).

![Figure 9.1: An example that not choosing any vertex from layer 1 can give advantage. Note that the triangles are used to denote terminals.](image)

From the claim, we therefore have (LP-3), which has polynomial number of constraints, as opposed to (LP-1) and (LP-2). Using the arguments similar to the proof of Claim 21 and Claim 20, one can show that (LP-1), (LP-2) and (LP-3) are equivalent on the tree instance, i.e. a solution to one LP can be transformed into another without increasing the LP cost.
\begin{align*}
\text{(LP-3)} & \quad \min \quad k \\
\text{s.t.} & \quad \sum_{v \in L_\tau} x(v) \leq k \quad \forall \tau : 1 \leq \tau \leq \lambda \\
& \quad \sum_{v \in P_t} x(v) \geq 1 \quad \forall t \in T \\
& \quad x(v) \geq 0 \quad \forall v \in V
\end{align*}

9.4 Our Results

We study the integrality gap of natural LP relaxation for the problem in general graphs, directed layered graphs, and trees. We use (LP-1) for general graphs and (LP-2) for the other two cases. Our results are summarized in Table 9.1. We note that all our upper bound results give polynomial time approximation algorithms that achieve those factors.

<table>
<thead>
<tr>
<th>Graph Classes</th>
<th>Previous LB</th>
<th>Previous UB</th>
<th>Our LB</th>
<th>Our UB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trees</td>
<td>2</td>
<td>$O(\log n)$</td>
<td>$\Omega(\log^* n)$</td>
<td>$O(\log^* n)$</td>
</tr>
<tr>
<td>Directed Layered Graphs</td>
<td>$\Omega(\log n)$</td>
<td>-</td>
<td>-</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>General Graphs</td>
<td>$\Omega(\log n)$</td>
<td>-</td>
<td>$n^{1/6}$</td>
<td>$O(n^{1/3} \log n)$</td>
</tr>
</tbody>
</table>

*Table 9.1: Summary of Our Results*

We remark that independently of our work, Anshelevich et. al. [9] studied RMFC and related problems. In particular, they give an $O(\sqrt{n})$-approximation LP-rounding algorithm for RMFC on general graphs. They also show that the integrality gap of the LP they are using is $\Omega(\log n)$, and give an $O(\log \ell)$-approximation for directed layered graphs with $\ell$ layers.
9.5 Organization

We start by presenting algorithmic results for all graph classes in Chapter 10. Then we show (essentially) matching lower bounds on the LP integrality gap for these graph classes in Chapter 11. Finally, we conclude with open problems in Chapter 12.
CHAPTER 10
APPROXIMATION ALGORITHMS

In this chapter, we present our algorithmic results.

10.1 Trees

In this section, we present our main result of this part: an $O(\log^* n)$ approximation algorithm for RMFC on trees. We start by giving a high-level overview in Section 10.1.1.

10.1.1 High-Level Overview

We now provide a high level overview of the ideas and techniques used in our $O(\log^* n)$-approximation algorithm for RMFC on trees. We root the tree at the vertex $s$ where the fire starts, and say that vertex $v$ lies in layer $\tau$ iff its distance from $s$ is $\tau$. Recall that we can assume w.l.o.g. that any solution chooses to save, at each time step $\tau$, a subset of vertices lying in layer $\tau$. With this observation, a natural LP relaxation for the problem assigns LP-weight $x(v)$ to each vertex $v$, with the constraints that the total LP-weight of vertices in any layer is at most $k$, the total LP-weight of any root-to-terminal path is at least 1, and the objective function of minimizing $k$. The main technical part of our algorithm is a randomized procedure, which, given an instance of RMFC on a tree and a set $T$ of terminals, produces an almost-feasible near-optimal solution, in the following sense: the output consists of a collection of subsets $U_{\tau}$ of vertices to be saved at each time step $\tau$, with $|U_{\tau}| \leq O(k)$ for each $\tau$, and an additional set $R$ of at most poly log $|T|$ vertices, such that sets $\{U_{\tau}\}_{\tau}$ induce a feasible solution in the graph obtained by removing the vertices of $R$ from $G$. In other words, if we recursively obtain a feasible solution for the instance where the vertices of $R$ serve as terminals, then the union of this solution with the collection $\{U_{\tau}\}_{\tau}$ of sets induces a feasible solution to the original problem. It is then easy to get an $O(\log^* n)$ approximation using the above procedure as a subroutine. The input graph $G$ is partitioned into $g = O(\log^* n)$
sub-instances $H_1, H_2, \ldots, H_g$, such that each source-to-terminal path traverses every sub-instance in this order and has total LP-weight of at least $\Omega(1/g)$ inside each of them. We then apply the randomized rounding algorithm to tree $H_g$ with the original set $T$ of terminals, obtaining the sets $\{U^g_\tau\}$ and $R^g$, where $|R^g| \leq \text{poly log } |T|$. Set $R^g$ of vertices is then used to define a set $T_{g-1}$ of terminals for instance $H_{g-1}$, where each vertex $v \in R^g$ has an ancestor in $T_{g-1}$. The randomized rounding procedure is then applied to $(H_{g-1}, T_{g-1})$, obtaining sets $\left(\{U^{g-1}_\tau\}, R^{g-1}\right)$, and so on. Since the size of the terminal set decreases fast with each iteration, with $|T_{i-1}| \leq \text{poly log } |T_i|$, set $T_1$ only contains a constant number of terminals, which are then added to the solution, together with sets $\left(\bigcup_{h=1}^{g} U^h_\tau\right)$. We note that in general, once the terminal set $T_h$ becomes small, say $|T_h| \leq O(\log \log n)$, the problem can be solved efficiently via exhaustive search. However, the vertices of $T_h$ do not necessarily belong to the original set $T$ of terminals, and so we are not guaranteed that there is a near-optimal solution for the instance in which $T_h$ is the set of terminals. Our LP-rounding algorithm proves that this indeed is the case, for the approximation factor of $O(\log^* n)$, while also providing a direct algorithm to find such a solution.

We now proceed to describe the randomized rounding procedure and its analysis, the main technical part of the paper. Observe first that the standard randomized rounding algorithm gives an $O(\log n)$-approximate solution for RMFC on trees, as follows: scan the layers from first to last, randomly selecting $k$ vertices in each layer, with probabilities proportional to their LP-values. It is easy to show that this process disconnects a constant fraction of terminals from the root with high probability. Repeating this procedure $O(\log n)$ times gives a feasible $O(\log n)$-approximate solution. This scheme is however somewhat wasteful: if vertex $v$ is selected to be in the solution, then there is no need to apply randomized rounding to its descendants, as the fire cannot reach any terminal in its subtree. Instead we can transfer the LP-weight from the descendants of $v$ to other vertices, thus increasing the probability of choosing the remaining vertices and hence covering their descendants by the randomized rounding procedure. The first step of our algorithm is transforming the solution
into a $1/M$ integral one, for $M = O(\log |T|)$, by a standard randomized rounding procedure. We then process the layers of the tree from first to last. At each step of the algorithm, we say that vertex $u$ survives iff none of its ancestors has been added to the solution so far. Consider some surviving vertex $v$ in some layer $L_\tau$ with positive LP-weight of, say, $x_v = 1/M$. Assume that the total LP-weight on the path connecting $v$ to the root is $h/M$. Then we select $v$ to be in the solution with probability roughly $1/(M - h)$. The resulting solution is guaranteed to be feasible, since for any path $P$ connecting a terminal to the root, at least one vertex of $P$ is in the solution (in particular, the last vertex of $P$ with non-zero LP-value is added to the solution with probability 1 if none of its ancestors belongs to it). The main technical part of the analysis is to show that the cost of the obtained solution is $O(k)$ with high probability. This is done by introducing a weight transfer mechanism: a way to transfer LP-weight from vertices whose ancestors are already in the solution to the remaining vertices. The weight transfer mechanism must on one hand ensure that the LP weight of every layer does not increase, while on the other hand, the LP-weight of each surviving vertex grows fast enough. We are unable to find a perfect weight transfer mechanism (in fact the lower bound on the integrality gap shows that it does not exist). Instead we add, throughout the algorithm, ancestors of vertices whose LP weight does not grow fast enough, to the set $R$. The heart of the analysis is designing a weight transfer scheme with the desired properties, while ensuring that $|R| \leq \text{poly log } |T|$.

### 10.1.2 The Algorithm

We use a parameter $g = O(\log^* n)$, and our final approximation factor is $O(g)$. We can assume, w.l.o.g., that $x(v) \leq 1/(2g)$: otherwise, we can add all vertices $v$ with $x(v) > 1/2g$ to the solution, remove their sub-trees from $G$ and solve the remaining instance. This will only increase the approximation ratio by at most a factor of 2. Our first step is to partition the tree $G$ into a collection of $g$ subgraphs $H_1, \ldots, H_g$, such that any root-to-terminal path in $G$ traverses $H_1, \ldots, H_g$ in this order and has an LP-weight of roughly $1/g$ inside each.
First, we partition the set $V$ of vertices into $g$ subsets $V_1,\ldots,V_g$, as follows. Add $s$ to $V_1$ and scan the remaining vertices in the non-decreasing order of their distance from $s$, breaking ties arbitrarily. Let $v$ be the current vertex, and assume that the parent of $v$ belongs to $V_j$. If $\sum_{u \in P \cap V_j} x(u) < 1/(2g)$, then $v$ is added to $V_j$, otherwise, it is added to $V_{j+1}$. It is easy to see that every root-to-terminal path $P$ in $G$ traverses $V_1,V_2,\ldots,V_g$ in this order, and for all $j$, $1/(2g) \leq \sum_{u \in P \cap V_j} x(u) \leq 1/g$. To construct graph $H_j$, we take the subgraph of $G$ induced by $V_j$, and if $j > 1$, we turn it into a tree by adding a source vertex $s_j$ that connects to every vertex $v \in V_j$ whose parent does not belong to $V_j$. We think for now of the leaves of $H_j$ as being the terminals in the corresponding RMFC instance. The LP-solution $x_j$ associated with $H_j$ is defined as follows: For $v \in V_j$, $x_j(v) = 2gx(v)$. Notice that the LP-weight of any root-to-leaf path inside $H_j$ is at least 1.

We will be solving each instance $H_j$ separately. Since $H_j$ is a tree, it can also be viewed as a layered graph. The layering of vertices in $H_j$ may be different from the original one, e.g. the first layer of $H_j$ may contain vertices from various subsets $L_\tau$. However, while solving instance $H_j$, we need to keep track of the original layering of its vertices. This motivates the notation we now introduce. Let $X^j_\tau$ denote the total LP-weight of vertices in $L_\tau$ in the LP-solution for $H_j$, $X^j_\tau = \sum_{v \in L_\tau \cap V_j} x_j(v)$. Recall that for each layer $L_\tau$, $\sum_{j=1}^{g} X^j_\tau \leq 2gk$, due to the scaling of LP-values $x(v)$ by the factor of $2g$. Given any vertex $v \in V_j$, let $P^j_v$ denote the path connecting $v$ to $s_j$ in graph $H_j$. Let $T_j$ be any subset of leaves of the tree $H_j$. We say that a subset $U^j \subseteq V_j$ of vertices is a feasible solution, or a feasible cut, for instance $(H_j,T_j)$ iff for each $t \in T_j$, at least one vertex on path $P^j_t$ belongs to $U^j$. We define, for each $1 \leq \tau \leq \lambda$, $U^j_\tau = U^j \cap L_\tau$ to be the subset of vertices of $U^j$ contained in layer $\tau$ of $G$.

We are now ready to present the high-level idea of our algorithm. We start by solving the sub-instance $(H_g,T_g)$, where $T_g$ is a subset of leaves of $H_g$, containing one ancestor for each terminal $t \in T$. Our goal is to produce a feasible solution $U^g$, where for each $1 \leq \tau \leq \lambda$, $|U^g_\tau| \leq O(X^g_\tau)$ (in fact we will be using amortized costs). Instead of finding such a solution,
we will find a partial solution: sets \( U^g, R^g \) such that \( U^g \cup R^g \) is a feasible solution for \((H_g, T_g)\), \(|R^g| \leq \text{poly log } |T_g|\), and the amortized cost of solution \( U^g \) is low. While \(|R^g|\) is relatively small, it is possible that many vertices of \( R^g \) lie in the same layer \( L_\tau \), while \(|R^g| \gg k\), so we cannot output \( U^g \cup R^g \) as the final solution, as its cost may be too high. Instead, we consider the instance \( H_{g-1} \), together with terminal set \( T_{g-1} \), containing, for each vertex \( v \in R^g \), the unique leaf of \( H_{g-1} \) lying on \( P_v \). We then continue the same process on \( H_{g-1} \), obtaining an even smaller set \( R^{g-1} \) and so on. The main technical part of the algorithm is summarized in the following theorem.

**Theorem 39** For each \( 1 \leq j \leq g \), for each subset \( T_j \) of leaves of \( H_j \), there is an efficient algorithm for finding sets \( U^j, R^j \subseteq V_j \) of vertices, such that:

- \(|R^j| \leq O(\log^5 |T_j|)\).
- \( U^j \cup R^j \) is a feasible solution for instance \((H_j, T_j)\), that is, for each \( t \in T_j \), \( P_t \cap (R^j \cup U^j) \neq \emptyset \).
- For all \( 1 \leq \tau \leq \lambda \), \( \sum_{\tau'=1}^{\tau} |U^j_{\tau'}| \leq O(\tau + \sum_{\tau'=1}^{\tau} X^j_{\tau'}) \) w.h.p., where \( U^j_{\tau} = U^j \cap L_\tau \).

We defer the proof of this theorem to the next section. We now show that it implies an \( O(g) \)-approximation algorithm. Apply Theorem 39 to graphs \( H_g, H_{g-1}, \ldots, H_1 \) in this order. In iteration \( i \), Theorem 39 is applied to graph \( H_{g-i+1} \) together with terminal set \( T_{g-i+1} \), computed as follows. For \( i = 1 \), \( T_g \) contains, for each terminal \( t \in T \), the unique leaf of \( H_g \) lying on \( P_t \). For \( i > 1 \), let \( R^{g-i+2} \) be the set in the output of Theorem 39, computed in iteration \((i-1)\). For each vertex \( v \in R^{g-i+2} \), let \( v' \) be the unique leaf of tree \( H_{g-i+1} \), lying on path \( P_v \) in \( G \). We then add \( v' \) to \( T_{g-i+1} \). Therefore, \(|T_{g-i+1}| \leq |R^{g-i+2}| \leq O(\log^5 |T_{g-i+2}|)\). We now use the following simple claim.

**Claim 22** For a suitably chosen \( g = O(\log^* n) \), \(|T_1| \) is bounded by a constant.
Proof: For $1 \leq i \leq g$, denote $Z_i = |T_{g-i+1}|$. We then have the following recursion: $Z_1 = |T_g| \leq n$, and for all $i > 1$, $Z_i \leq c \cdot \log^5 Z_{i-1}$, for some constant $c > 1$. We need to prove that $Z_g$ is bounded by a constant for a suitably chosen $g = O(\log^* n)$.

We prove by induction that $Z_i \leq (10c\log^{(i-1)} n)^5$. The base case when $i = 1$ holds trivially. Assume that the statement holds for $i$ and consider $Z_{i+1}$. Then $Z_{i+1} \leq c \log^5 Z_i \leq c \left(\log \left( (10c\log^{(i-1)} n)^5 \right) \right)^5$, by the induction hypothesis. We then get that

$$Z_{i+1} \leq c \left(5\log(10c) + 5\log(i) n\right)^5 \leq \left(10c\log(i) n\right)^5$$

whenever $\log(i) n > 5c\log(10c)$. So for $g = O(\log^* n)$, $Z_g$ is bounded by a constant. \qed

The final solution is $U = \left( \bigcup_{j=1}^{g} U_j \right) \cup T_1$. For each $\tau : 1 \leq \tau \leq \lambda$, let $U_\tau = U \cap L_\tau$. Theorem 39 ensures that $\sum_{\tau' \leq \tau} |U_{\tau'}| = \sum_{j=1}^{g} \sum_{\tau' \leq \tau} |U^j_{\tau'}| + |T_1| \leq \sum_{j=1}^{g} O \left( \tau + \sum_{\tau' \leq \tau} X^j_{\tau'} \right) + O(1) \leq O(g\tau) + O(g\tau\text{OPT}) + O(1) = O(g\tau\text{OPT})$. Therefore, the amortized cost of the solution $U$ is $O(g \cdot \text{OPT})$. It is easy to see that $U$ is a feasible set, as every terminal $t$ has at least one vertex $v \in P_t \cap U$.

### 10.1.3 Proof of Theorem 39

We start with the tree $H_j$ and a subset $T_j$ of leaves of $H_j$, that we call terminals. Since from now on we focus on a specific tree $H_j$, to simplify the notation we will omit the index $j$. So we denote $H_j$ by $H$, $V_j$ by $V$, and for all $1 \leq \tau \leq \lambda$, $L^j_\tau = V_j \cap L_\tau$ is denoted by $L_\tau$. We denote $T_j$ by $T$, and the total number of terminals is denoted by $N = |T|$. The LP-solution $x_j(v)$ is denoted by $x(v)$ and $X^j_\tau$ is denoted by $X_\tau$, so $X_\tau = \sum_{v \in L_\tau} x(v)$. We also use the fractional amortized cost, $\overline{X}_\tau = \sum_{\tau' \leq \tau} X_{\tau'}$ in our analysis. For a vertex $v \in V$, let $P_v$ be the unique path connecting $v$ to the root $s$ of $H$. Recall that for each $t \in T$, $\sum_{v \in P_t} x(v) \geq 1$. Our goal is to find two sets $U, R \subseteq V$ of vertices, such that $|R| = O(\log^5 N)$, and for each terminal $t \in T$, $P_t \cap (R \cup U) \neq \emptyset$. For each $1 \leq \tau \leq \lambda$, denote $U_\tau = U \cap L_\tau$. We also need to ensure that the amortized cost $\sum_{\tau' = 1}^\tau |U_{\tau'}| \leq O(\tau + \overline{X}_\tau)$ for all $\tau$. We use a parameter $M$, the
smallest power of 2 whose value is greater than $100\log N$, so $M = \Theta(\log N)$. The algorithm has three steps. In the first step we transform the fractional solution into a $1/M$-integral one, with only $O(N^2)$ vertices having non-zero LP-weight.

In the second step we partition the vertices with non-zero LP-weight into subsets $F_1, F_2, \ldots, F_M$. In the last step we compute the set $R$ and perform randomized rounding to obtain set $U$.

**Step 1: Obtaining a $1/M$-integral solution**

Given any fractional solution $y$, let $Y_\tau = \sum_{v \in \mathcal{L}_\tau} y(v)$ and $\overline{Y}_\tau = \sum_{\tau' \leq \tau} Y_{\tau'}$ for all $1 \leq \tau \leq \lambda$. This step is summarized in the next theorem whose proof uses standard randomized rounding techniques.

**Theorem 40** Given any feasible fractional solution $x$ for instance $(H, \mathcal{T})$, there is an efficient randomized algorithm to find a $(1/M)$-integral feasible solution $y$, such that the number of vertices $v$ with $y(v) > 0$ is at most $N^2$, and for all $\tau \geq 1$, $\overline{Y}_\tau \leq O(\tau + \overline{X}_\tau)$. The algorithm succeeds with probability at least $1 - 2/N^2$.

**Proof:** For each terminal $t \in \mathcal{T}$, let $P_t$ denote the unique path connecting $t$ to the root of $H$. We start by removing from $H$ all vertices that do not lie on any path $P_t$ for $t \in \mathcal{T}$. Let $H' = (V', E')$ be the resulting tree and $\mathcal{L}'_1, \ldots, \mathcal{L}'_\lambda$ be the resulting layers, so $\mathcal{L}'_\tau = V' \cap \mathcal{L}_\tau$ for all $\tau$. Observe that since every vertex of $V'$ lies on some path $P_t$ for $t \in \mathcal{T}$, $|\mathcal{L}'_\tau| \leq N$ for all $\tau$. Next, for each vertex $v \in V'$ whose distance from the root is at most $N$, we set $x'(v) = x(v) + 1/N$, and for all other vertices we set $x'(v) = 0$. Since $H'$ is a tree with $N$ leaves, the number of vertices with non-zero value $x'(v)$ is now at most $N^2$. It is easy to see that $x'$ is a feasible fractional solution, as the summation of values $x'(v)$ along any path $P_t$ for $t \in \mathcal{T}$ is at least 1: If $P_t$ contains at least $N$ vertices, then values $x'(v)$ of the first $N$ vertices of $P_t$ are at least $1/N$, and their total sum is at least 1. Otherwise, if $P_t$ contains less than $N$ vertices, then for each vertex $v \in P_t$, $x'(v) \geq x(v)$, so the total sum remains at least 1. For each $1 \leq \tau \leq \lambda$, let $X'_\tau = \sum_{v \in \mathcal{L}'_\tau} x'(v)$, and let $\overline{X}'_\tau = \sum_{\tau' \leq \tau} X'_{\tau'}$. Since
$|L'_\tau| \leq N$, we have that $X'_\tau \leq X_\tau + 1$ and $X'_\tau \leq \tau + \lambda$ for all $\tau$.

The next step is to perform randomized rounding. Let $z$ be the largest index for which $X'_z \leq 1$. For vertices $v \in L_\tau$ with $\tau \leq z$, we set $y(v) = 0$. Consider now any vertex $v \in L'_\tau$ for $\tau > z$. We can write $x'(v) = \frac{c_v}{M} + p_v$ where $p_v < 1/M$ and $c_v = \left\lfloor x'(v)M \right\rfloor$. We set $y(v) = \frac{c_v + b_v}{M}$, where $b_v = 1$ with probability $Mp_v$, and it is 0 otherwise. We first show that the resulting solution $y$ is “almost feasible” w.h.p.:

**Claim 23** With probability at least $1 - 1/N^2$, for each terminal $t \in T$, $\sum_{v \in P_t} y(v) \geq 1/8$.

**Proof:** Fix some terminal $t \in T$. Let $P'_t \subseteq P_t$ contain all vertices lying in layers $L'_\tau$ with $\tau > z$. Since the total weight $X'_z$ of all vertices in layers $L'_1, \ldots, L'_z$ is bounded by $1/2$, $\sum_{v \in P'_t} x'(v) \geq 1/2$. We now consider two cases. If $\sum_{v \in P'_t} c_v/M \geq 1/4$, then clearly $\sum_{v \in P'_t} y(v) \geq 1/4$. Otherwise, $\sum_{v \in P'_t} p_v \geq 1/4$. Consider the random variables $b_v$ for $v \in P'_t$. These are independent $\{0, 1\}$ variables with $\mathbb{E} \left[ \sum_{v \in P'_t} b_v \right] = M \sum_{v \in P'_t} p_v \geq M/4$.

Using Chernoff bound:

$$\Pr \left[ \sum_{v \in P'_t} y(v) \leq 1/8 \right] \leq \Pr \left[ \sum_{v \in P'_t} b_v \leq M/8 \right] \leq e^{-M/32}$$

Using the union bound over all terminals, and the fact that $M \geq 100 \log N$, we get that with probability at least $1 - 1/N^2$, for each terminal $t \in T$, $\sum_{v \in P_t} y(v) \geq 1/8$. \qed

We now bound the amortized cost of the solution $y$.

**Claim 24** With probability at least $1 - 1/N^2$, for each $\tau : 1 \leq \tau \leq \lambda$, $Y_\tau \leq 8X'_\tau$.

**Proof:** The proof is again a simple application of the Chernoff bound. Let $I$ be the set of indices $\tau$ for which $L'_\tau$ contains at least one vertex $v$ with non-zero value $x'(v)$. Recall that $|I| \leq N^2$. It is enough to prove that the claim holds for all $\tau \in I$, since for $\tau \notin I$, $Y_\tau = 0$.

For $\tau \leq z$, $Y_\tau = 0$ so the claim clearly holds.

Consider now some $\tau \in I$, $\tau > z$. Let $S$ contain all vertices $v \in L'_\tau$ for all $\tau' : z < \tau' \leq \tau$, so $Y_\tau = \sum_{v \in S} y(v)$. Let $C_\tau = \sum_{v \in S} c_v$ and $P_\tau = \sum_{v \in S} p_v$, so $X'_\tau \geq C_\tau/M + P_\tau$. Let
Let $v \in V$ be any vertex with $y(v) = r/M$ for some integer $r > 0$, and assume that $\sum_{u \in P_v \setminus \{v\}} y(u) = h/M$, for some integer $h \geq 0$. Then $v$ belongs to sets $F_{h+1}, F_{h+2}, \ldots, F_{h+r}$.

We view the weight $y(v)$ of $v$ as being evenly split among the $r$ sets, and so the weight of $v$ with respect to $F_{h+r'}$ is $y_{h+r'}(v) = 1/M$, for all $1 \leq r' \leq r$. We then have that $\sum_{h:v \in F_h} y_h(v) = y(v)$. For each terminal $t \in T$, for each $h : 1 \leq h \leq M$, there is exactly one vertex $v \in P_t \cap F_h$. We now group sets $F_1, \ldots, F_M$ geometrically, by defining $\log M$ sets of indices $I_1, \ldots, I_{\log M}$ (recall that $M$ is a power of 2). Set $I_1$ contains the first $M/2$ indices, set $I_2$ contains the next $M/4$ indices, and so on, with the last set $I_{\log M}$ containing a single index $M$. Formally: $I_q = \left\{ b+r \mid b = \sum_{1 \leq q' < q} M/2^{q'}, 1 \leq r \leq M/2^q \right\}$.

**Step 3: Randomized Rounding**

This is the main step of the algorithm. The goal is to choose a set $U$ of vertices that will serve as the output. We will also define a set $R$ of vertices, $R = \bigcup_{h=0}^{M} R_h$. The algorithm has $M$ iterations. Set $R_0$ is selected before the first iteration starts (we specify its choice below), and for each $h : 1 \leq h \leq M$, set $R_h$ is selected in iteration $h$.

\[ B_\tau = \sum_{v \in S} b_v, \text{ so } Y = (C_\tau + B_\tau)/M. \] Recall that $X'_\tau \geq \frac{1}{2}$ for $\tau > z$.

We now consider two cases. First, if $P_\tau \geq \frac{1}{2}$, then $E[B_\tau] = MP_\tau \geq M/2$. Therefore, using Chernoff bound, $\Pr[B_\tau \geq 6MP_\tau] \leq 2^{-3M}$. But if $B_\tau \leq 6MP_\tau$, then $\overline{Y}_\tau = \frac{C_\tau + B_\tau}{M} \leq \frac{C_\tau + 6P_\tau M}{M} \leq 6 \left( \frac{C_\tau}{M} + P_\tau \right) \leq 6X'_\tau$.

Assume now that $P_\tau < \frac{1}{2}$. Then $E[B_\tau] \leq M/2$. Again, using Chernoff bound, $\Pr[B_\tau \geq 3M] \leq 2^{-3M}$. But if $B_\tau \leq 3M$, then $\overline{Y}_\tau = \frac{C_\tau + B_\tau}{M} \leq \frac{C_\tau}{M} + 3 \leq 8X'_\tau$, since $X'_\tau \geq 1/2$. We then have that $\overline{Y}_\tau \leq 8X'_\tau$ with probability at least $1 - 2^{-3M}$. Using the union bound over all $\tau \in I$ and the fact that $M \geq 100 \log N$ while $|I| \leq N^2$ gives the desired result. \hfill \Box

The final solution is obtained by scaling the values $y(v)$ up by the factor of 8. \hfill \Box

Let $E_1$ be the event that the algorithm in Theorem 40 succeeds, so $\Pr[E_1] \geq 1 - 2/N^2$.

**Step 2: Defining Sets $F_1, \ldots, F_M$.**
Consider the beginning of iteration $h$ and assume that $h \in I_q$ for some $1 \leq q \leq \log M$. Vertex $v \in F_h$ is called active iff no vertex of $P_v$ currently belongs to $R \cup U$. Let $A_h \subseteq F_h$ denote the subset of vertices of $F_h$ that are active at the beginning of iteration $h$. We select a subset $R_h \subseteq A_h$ of vertices to be added to $R$ (we show how to select this subset below).

Next, each vertex $v \in A_h \setminus R_h$ is added to $U$ with probability $\min\{16 \cdot 2^q / M, 1\}$.

This completes the description of the algorithm, except for the definition of the sets $R_h$. It will be convenient to partition the algorithm’s execution into $\log M$ phases, where phase $q$ consists of iterations $h \in I_q$. We now define $U_\tau = U \cap \mathcal{L}_\tau$ for all $1 \leq \tau \leq \lambda$. It is easy to see that $U \cup R$ is a feasible solution for $(H, \mathcal{T})$, since for every terminal $t \in \mathcal{T}$, $P_t \cap (U \cup R) \neq \emptyset$ (otherwise, the vertex $v \in P_t \cap F_M$ should have been added to $U$ with probability 1).

It is easy to see that the expected amortized cost of solution $U$ for each $\tau$ is low. In particular, if we show that each vertex $v \in F_h$ is added to $U$ with probability at most $O(y_h(v))$, then the expected amortized cost of $U$ for each $\tau$ is $O(\overline{Y_\tau})$: Consider some $v \in F_h$ for some $h \in I_q$. By the definition of the decomposition $F_1, F_2, \ldots, F_M$, vertex $v$ has one ancestor $v_{h'} \in F_{h'}$ for all $h' < h$. Vertex $v$ is added to set $U$ iff it remains active until iteration $h$, and it is chosen by the randomized rounding procedure. Consider some $q' < q$. The probability that no vertex in set $\{v_{h'} : h' \in I_{q'}\}$ is added to $U$ is bounded by $(1 - 2^{q'+4}/M)^M/2^{q'} \leq 1/2$, so the probability that $v$ remains active at the beginning of iteration $h$ is at most $1/2^{q-1}$. Since $v$ is selected by the randomized rounding procedure with probability at most $16 \cdot 2^{q}/M$, overall $v$ is added to $U$ with probability at most $O(y_h(v)) = O(1/M)$. Therefore, the expected amortized cost of $U$ for every $\tau$ is $O(\overline{Y_\tau})$. This bound on the expectation is however not enough for us, and is only provided here for intuition. We need to prove that the amortized cost of the set $U$ is low for each $\tau$ with high probability; the proof of this claim is more involved.

**Analysis.** Recall that if event $\mathcal{E}_1$ happens, then $\overline{Y_\tau} \leq O(\tau + \overline{X_\tau})$ for all $1 \leq \tau \leq \lambda$. In order to bound the amortized solution cost, it is therefore enough to prove that with high probability:
∀τ : 1 ≤ τ ≤ λ \sum_{τ' ≤ τ} |U_{τ'}| \leq O(\overline{Y}_τ) \tag{10.1}

Recall that the partition \{L_τ\}_{τ=1}^λ of vertices of H corresponds to the original layers in graph G, while subsets \{F_h\}_{h=1}^M are defined according to the distribution of the LP-weight on vertices of H. We break the summation in (10.1) down by sets \{F_1, \ldots, F_M\} as follows.

For each 1 ≤ τ ≤ λ, for each 1 ≤ h ≤ M, let \mathcal{L}_{τ,h} = F_h \cap \mathcal{L}_τ and \mathcal{U}_{τ,h} = F_h \cap \mathcal{U}_τ. Let \overline{Y}_{τ,h} = \sum_{v \in \mathcal{L}_{τ,h}} y_h(v) and \overline{Y}_{τ,h} = \sum_{τ' \leq τ} Y_{τ',h}. So \sum_{h=1}^M \overline{Y}_{τ,h} \leq \overline{Y}_τ. In order to show that (10.1) holds w.h.p., it is now enough to show that w.h.p.:

∀ h : 1 ≤ h ≤ M, ∀ τ : 1 ≤ τ ≤ λ \sum_{τ' ≤ τ} |U_{τ',h}| \leq 200\overline{Y}_{τ,h} \tag{10.2}

For each τ : 1 ≤ τ ≤ λ, let \mathcal{A}_{τ,h} contain the vertices of \mathcal{F}_h \cap \mathcal{L}_τ, that are active at the beginning of iteration h, i.e. \mathcal{A}_{τ,h} = A_h \cap \mathcal{L}_τ. We say that event \mathcal{E}_2(h) holds for \textbf{i}f:

∀ τ : 1 ≤ τ ≤ λ \sum_{τ' ≤ τ} |A_{τ',h}| \cdot \frac{2^q}{M} \leq 2\overline{Y}_{τ,h} \tag{10.3}

Let \mathcal{E}_2 = \bigcap_{1 ≤ h ≤ M} \mathcal{E}_2(h). The heart of our algorithm and its analysis is to show the choice of sets \mathcal{R}_h, for which events \mathcal{E}_2(h) happen with high probability for all h. The formal statement is summarized in the following theorem:

**Theorem 41** There is an efficient procedure for selecting, in each iteration \textbf{h} : 1 ≤ h ≤ M, a subset \mathcal{R}_h \subseteq \mathcal{A}_h of vertices, with |\mathcal{R}_h| \leq O(\log^4 N), such that for each h : 1 ≤ h ≤ M, event \mathcal{E}_2(h) happens with probability at least 1 − 1/N^2.

The proof of Theorem 41 appears in the next section. We first complete the proof of Theorem 39 using Theorem 41. If event \mathcal{E}_2(h) happens, then \sum_{τ' ≤ τ} |A_{τ',h}| \cdot \frac{2^q}{M} \leq 2\overline{Y}_{τ,h}. In iteration h, our algorithm chooses, for each τ, vertices in \mathcal{A}_{τ,h} \setminus \mathcal{R}_h to be added to \mathcal{U}_{τ,h} with probability \textbf{m}in \{16 \cdot 2^τ/M, 1\} each. So intuitively, if (10.3) holds, then from Chernoff
bound, (10.2) also holds with high probability for all $\tau$ and $h$. This approach indeed works, except for the cases where $\mathcal{Y}_{\tau,h}$ is small, and so the probability of success guaranteed by the Chernoff bound is low. We take care of this problem by adding vertices to set $R_0$ as follows. Fix some $h : 1 \leq h \leq M$. Let $\tau_h$ be the largest index for which $\mathcal{Y}_{\tau_h,h} \leq M$. Add all the vertices in $\bigcup_{\tau \leq \tau_h} \mathcal{L}_{\tau,h}$ to $R_0$. Notice that since every vertex in $F_h$ has LP-value $y_h(v) \geq 1/M$, we add at most $M^2$ vertices for each $h$, so $|R_0| \leq M^3$. It is now easy to complete the proof of Theorem 39. Let $\mathcal{E}_3(h)$ be the event that (10.2) holds for $h$, and let $\mathcal{E}_3 = \bigcap_{1 \leq h \leq M} \mathcal{E}_3(h)$. The proof of the next theorem follows from applying standard Chernoff bounds.

**Theorem 42** Event $\mathcal{E}_3$ happens with probability at least $1 - 1/N$.

**Proof:** We start with the following claim:

**Claim 25** For each $h : 1 \leq h \leq M$, we have that $\Pr[\neg \mathcal{E}_3(h) \mid \mathcal{E}_1 \land \mathcal{E}_2(h)] \leq 1/N^2$.

**Proof:** Assume that $h \in I_q$ and consider some $\tau : 1 \leq \tau \leq \lambda$. If $\tau \leq \tau_h$, then $U_{\tau,h} = \emptyset$ for each $\tau' \leq \tau$, so Equation (10.2) holds for this choice of $\tau, h$. Otherwise, if $\tau > \tau_h$, then since we condition on event $\mathcal{E}_2(h)$, $\sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot \frac{2^q}{M} \leq 2\mathcal{Y}_{\tau,h}$. For each $v \in \left( \bigcup_{\tau' \leq \tau} A_{\tau',h} \right) \setminus R_h$, let $Z_v$ be the indicator variable for including $v$ in $U$, so $\sum_{\tau' \leq \tau} |U_{\tau',h}| = \sum_{\tau' \leq \tau} \sum_{v \in A_{\tau',h} \setminus R_h} Z_v$. Then $\mathbb{E} \left[ \sum_{\tau' \leq \tau} \sum_{v \in A_{\tau',h} \setminus R_h} Z_v \right] \leq 16 \sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot \frac{2^q}{M} \leq 32\mathcal{Y}_{\tau,h}$. Using the Chernoff bound, $\Pr \left[ \sum_{\tau' \leq \tau} |U_{\tau',h}| > 200\mathcal{Y}_{\tau,h} \right] \leq 2^{-200\mathcal{Y}_{\tau,h}} \leq 2^{-200M}$, since $\mathcal{Y}_{\tau,h} \geq M$. Applying the union bound over all values $\tau : \tau_h \leq \tau \leq \lambda$ for which $\mathcal{L}_{\tau,h} = \emptyset$ (notice that there are at most $N^2 \cdot M$ such values), we get the desired result.

Since

$$
\Pr[\neg \mathcal{E}_3(h)] \leq \Pr[\neg \mathcal{E}_3(h) \mid \mathcal{E}_1 \land \mathcal{E}_2(h)] + \Pr[\neg (\mathcal{E}_1 \land \mathcal{E}_2(h))] \\
\leq \Pr[\neg \mathcal{E}_3(h) \mid \mathcal{E}_1 \land \mathcal{E}_2(h)] + 3/N^2
$$

we get that $\Pr[\neg \mathcal{E}_3(h)] \leq 4/N^2$. Using the union bound over all indices $h : 1 \leq h \leq M$
finishes the proof.

Finally, we set \( R = \bigcup_{h=0}^{M} R_h \). Since \( |R_h| \leq O(\log^4 N) \) for all \( h \), we have that \( |R| \leq O(\log^5 N) \) as required. This completes the proof of Theorem 39, except for the proof of Theorem 41.

### 10.1.4 Proof of Theorem 41

We start with a high-level overview of the proof. For a vertex \( v \in H \), let \( L(v) \) be the layer \( L^\tau \) to which \( v \) belongs. Consider the set \( F_h \) for some fixed \( h \in I_q \), for some \( 1 \leq q \leq \log M \), and the partition \( F_h = \bigcup_{\tau=1}^{\lambda^h} L_{\tau,h} \) of set \( F_h \) into layers, where \( L_{\tau,h} = F_h \cap L^\tau \). We say that \( v \in F_h \) is active at the beginning of iteration \( h' < h \) iff no vertex of \( P_v \) has been added to \( U \cup R \) in iterations 1, 2, \ldots, \( h' - 1 \). Let \( A^h_{h'} \subseteq F_h \) denote the set of active vertices of \( F_h \) at the beginning of iteration \( h' \), so \( A^1_{h'} = F_h \setminus R_0 \), and let \( A^{h'}_{\tau,h} = A^h_{h'} \cap L^\tau \). Similarly, for each phase \( q' : 1 \leq q' \leq q \), we denote by \( A^{(q')}_{h'} \subseteq F_h \) the set of vertices that are active at the beginning of phase \( q' \). Since the LP-weight of each vertex \( v \in F_h \) w.r.t. \( F_h \) is \( y^h_{h'}(v) = \frac{1}{M} \), we have, for each \( \tau : 1 \leq \tau \leq \lambda \), \( |A^1_{\tau,h}| \cdot \frac{1}{M} \leq Y_{\tau,h} \), and \( \sum_{\tau' \leq \tau} |A^1_{\tau',h}| \cdot \frac{1}{M} \leq Y_{\tau,h} \) (the inequality is since \( A^1_{\tau,h} \) does not include the vertices of \( R_0 \), which are counted in \( Y_{\tau,h} \)). As the algorithm progresses, some of the vertices of \( F_h \setminus R_0 \) become inactive. Our main idea is to transfer the LP-weight from such vertices to the remaining active vertices of \( F_h \). The transfer is only performed from vertex \( v \in L^\tau \) to vertex \( v' \in L^{\tau'} \) if \( \tau \leq \tau' \). This ensures that throughout the algorithm, the amortized fractional weight \( Y_{\tau,h} \) does not increase for any \( \tau \). For each vertex \( v \in A^h_{h'} \), we denote by \( y^{h'}_{h'}(v) \) its LP-weight at the beginning of iteration \( h' \), and by \( y^{(q')}_{h'}(v) \) its LP-weight at the beginning of phase \( q' \). Our goal is to ensure that for each phase \( 1 \leq q' \leq q \), for each vertex \( v \in A^{(q')}_{h'} \) that remains active at the beginning of phase \( q' \), its LP-weight \( y^{(q')}_{h'}(v) \geq \frac{2^{q' - 1}}{M} \). Therefore, the LP-weight of the surviving active vertices doubles in each phase, and eventually, at the beginning of iteration \( h \), \( y^h_{h'}(v) \geq \frac{2^{q-1}}{M} \) for each active vertex \( v \). Since the amortized fractional LP-weight \( Y_{\tau,h} \) does not increase throughout the algorithm, this will give the desired bound \( \sum_{\tau' \leq \tau} |A^{h'}_{\tau',h}| \cdot \frac{2^{q'}}{M} \leq 2Y_{\tau,h} \) for
all \( \tau \). The main ingredient of our analysis is a \textit{weight transfer mechanism} for moving the LP-weight from vertices that become inactive to those that remain active in \( F_h \).

We now provide an informal overview of the weight transfer scheme. We focus on the weight transfer within a specific set \( F_h \), where \( h \in I_q \) for some \( 1 \leq q \leq \log M \). Let \( h' < h \) be the index of the current iteration, and \( h' \in I_{q'} \). Assume that at the beginning of phase \( q' \), the LP-weight of each active vertex \( v \in A_h^{(q')} \) is \( y_h^{(q')} (v) \geq \frac{2^{q'} - 1}{M} \). We need to ensure that the LP-weight of each surviving active vertex of \( F_h \) becomes at least \( \frac{2^{q'}}{M} \) at the end of phase \( q' \). Since phase \( q' \) has \( M/2^{q'} \) iterations, it is enough to ensure that the LP-weight of each surviving active vertex increases by at least \( (2^{q'}/M)^2 \) in each iteration \( h' \in I_{q'} \). Consider the following simple weight transfer mechanism: Fix some partition of the active vertices \( A_h^{h'} \subseteq F_h \) into subsets \( Z_1, \ldots, Z_r \) of roughly equal size. For each subset \( Z_j, 1 \leq j \leq r \), if some vertex \( v \in Z_j \) becomes inactive during iteration \( h' \), then its LP-weight is evenly split among the vertices of \( Z_j + 1 \). Observe that for each active vertex \( v \in A_h^{h'} \), the probability to become inactive in iteration \( h' \) is at least \( 16 \cdot \frac{2^{q'}}{M} \) (this is the probability that the ancestor of \( v \) in \( F_h \) is added to \( U \)). So we expect a \( (16 \cdot \frac{2^{q'}}{M}) \)-fraction of vertices of \( Z_j \) to become inactive. In case a close number of vertices (say, \( |Z_j| \cdot 8 \cdot \frac{2^{q'}}{M} \) vertices) become inactive, each vertex in \( Z_j + 1 \) will receive a total contribution of \( \frac{|Z_j| \cdot 8 \cdot 2^{q'}}{M} \cdot \frac{2^{q'} - 1}{M} \cdot \frac{1}{|Z_j| + 1} \geq \left( \frac{2^{q'}}{M} \right)^2 \), as desired. There are two main obstacles to this approach. First, in order to show that w.h.p., for each \( Z_j \), the number of vertices that become inactive is close to the expected one, we need to ensure that there is a certain degree of independence among these events (for example, if all vertices in \( Z_j \) have one common ancestor in \( F_{h'} \), this will not be the case). So ideally, we would like set \( Z_j \) to contain a roughly equal number of descendants of many (say \( \Omega(\log^2 N) \)) distinct vertices of \( A_h^{h'} \). At the same time, in order to guarantee that the amortized cost does not grow, we need to ensure that for each \( v \in Z_j \) and \( u \in Z_{j+1} \), \( L(v) \leq L(u) \) for all \( j \), so the weight is only transferred from layer \( L_\tau \) to layer \( L_{\tau'} \) where \( \tau \leq \tau' \). This motivates the weight transfer mechanism that we formally describe below.

We now turn to the formal proof of Theorem 41. The proof consists of two parts. In the
first part we define the procedure for selecting subsets \( R_{h'} \subseteq A_{h'}^{h'} \) of vertices to be added to the set \( R \) in each iteration \( h' \). The second part proves that for each \( h : 1 \leq h \leq M \), event \( \mathcal{E}_2(h) \) happens with high probability, under this choice of sets \( R_{h'} \). We start with the first part. Consider some iteration \( h' \) of the algorithm, \( 1 \leq h' \leq M \). For each \( h : h' < h \leq M \), we define a partition \( C^h_{i,j} \) of the set \( A_{h'}^{h'} \) of vertices. (Notice that each \( h > h' \) defines a distinct partition \( \{C^h_{i,j}\} \) of \( A_{h'}^{h'} \).) This partition is used, on the one hand, to define a subset \( R_{h',h} \subseteq A_{h'}^{h'} \) of vertices, and on the other hand, we later use it to define a weight transfer mechanism. Eventually we set \( R_{h'} = \bigcup_{h > h'} R_{h',h} \).

**Defining Sets \( C^h_{i,j} \) and set \( R_{h',h} \).** We now fix some index \( h : h' < h \leq M \) and define the partition \( \{C^h_{i,j}\} \) of \( A_{h'}^{h'} \) induced by \( F_h \). To simplify the notation, we denote the sets \( C^h_{i,j} \) by \( C_{i,j} \) here.

Consider first some vertex \( v \in A_{h'}^{h'} \) that is active at the beginning of iteration \( h' \). Let \( D(v) \subseteq A_{h'}^{h'} \) be the set of the descendants of \( v \) in \( F_h \) that are currently active. We order the vertices \( u \in D(v) \) in the non-decreasing order of layers \( L(u) \), \( D(v) = \{u_1, \ldots, u_{z_v}\} \), where \( L(u_1) \leq L(u_2) \leq \cdots \leq L(u_{z_v}) \). We now group the vertices of \( D(v) \) geometrically, with \( D_1(v) = \{u_1\}, D_2(v) \) containing the next two vertices and so on. Formally, for \( 1 \leq i \leq \lceil \log z_v \rceil \), \( D_i(v) = \{u_{b+r} \mid 1 \leq r \leq 2^{i-1}, b = \sum_{i' \leq i} 2^{i'-1}\} \), and \( D_{\lceil \log z_v \rceil}(v) \) contains the remaining vertices, \( D_{\lceil \log z_v \rceil}(v) = D(v) \setminus \left( \bigcup_{i < \lceil \log z_v \rceil} D_i(v) \right) \).

We now define \( \log N \) classes of vertices in \( A_{h'}^{h'} \), as follows: \( C_i = \left\{ v \in A_{h'}^{h'} : |D_i(v)| = 2^{i-1} \right\} \) (notice that vertex \( v \) belongs to \( \Theta(\log z_v) \) such classes). Observe that the set \( \bigcup_{v \in C_i} D_i(v) \) contains exactly \( 2^{i-1} \) descendants for each vertex \( v \in C_i \). We group the vertices of \( C_i \) into subsets, containing \( O(\log^2 N) \) vertices each. The descendants \( D_i(v) \) of vertices in each such subset will serve as the sets \( Z_j \) from the intuitive explanation above. Since we can only transfer LP-weight from layer \( \mathcal{L}_\tau \) to layer \( \mathcal{L}_{\tau'} \), where \( \tau \leq \tau' \), we need to perform this grouping carefully.

Fix some class \( C_i \), for \( 1 \leq i \leq \log N \). For each vertex \( v \in C_i \), consider the corresponding
set $D_i(v)$ of its $2^{i-1}$ descendants in $A_h^{h'}$. Let $\tau_i(v)$ be the largest index $\tau$ of a layer to which any vertex in $D_i(v)$ belongs, so $\tau_i(v) = \max_{u \in D_i(v)} \{L(u)\}$. We now order the vertices of $C_i$ in the non-decreasing order of $\tau_i(v)$, and partition them into consecutive subsets of $10M^2$ vertices each in this order. Formally, we partition set $C_i$ in the non-decreasing order of $\tau_i(v)$, where each set, except possibly for the last one, contains $10M^2$ vertices, and if $v \in C_{i,j}$ and $u \in C_{i,j+1}$ then $\tau_i(v) \leq \tau_i(u)$. This completes the definition of the partition $C_{i,j}$ of $A_h^{h'}$ induced by $F_h$. We denote the corresponding values $\alpha_i$, for $1 \leq i \leq \log N$ by $\alpha_i^{h',h}$, omitting the superscripts when clear from context. We now define $R_{h',h} = \bigcup_{i=1}^{\log N} C_i^{h'}$.

Observe that $|R_{h',h}| \leq O(\log N \cdot M^2) = O(\log^3 N)$. We set $R_{h'} = \bigcup_{h=h'+1}^M R_{h',h}$, and so $|R_{h'}| \leq O(\log^4 N)$. This completes the first part of the proof.

From now on we fix an index $h : 1 \leq h \leq M$, and assume that $h \in I_q$ for some $1 \leq q \leq M$. Our goal is to prove that event $E_2(h)$ happens with probability at least $1 - 1/N^2$ w.r.t. the choice of sets $R_{h'}$ described above. Consider some iteration $h' : 1 \leq h' < h$ of the algorithm, and the partition $\{C_{i,j}\}_{i,j}$ of $A_h^{h'}$ induced by $F_h$. Fix some set $C_{i,j}$ of the partition, for some $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$. Since each vertex in $C_{i,j}$ is added to $U \cup R_{h'}$ with probability at least $16 \cdot 2^{q'}/M$ in iteration $h'$ and $|C_{i,j}| = 10M^2$, we expect at least $160M \cdot 2^{q'}$ vertices of $C_{i,j}$ to be added to $U \cup R_{h'}$ in iteration $h'$. Let $E(h,h',i,j)$ be the good event that at least $80M \cdot 2^{q'}$ vertices of $C_{i,j}$ are added to $U \cup R_{h'}$. We now proceed as follows. First, we show that with high probability, events $E(h,h',i,j)$ happen for all relevant indices $h',i,j$. This is done in the next claim, whose proof uses standard Chernoff bound together with the union bound. Next we show a weight transfer procedure, which ensures that for each $1 \leq h' < h$, if $E(h,h',i,j)$ happens for all $i$ and $j$, then each remaining active vertex in $A_h^{h'}$ receives a sufficient contribution to its LP-weight.

**Claim 26** Events $E(h,h',i,j)$ hold for all $1 \leq h' < h$, $1 \leq i \leq \log N$, $1 \leq j < \alpha_i^{h',h}$ with probability at least $1 - 1/N^2$.

**Proof:** Since $\alpha_i^{h',h} \leq N$ for all $i,h,h'$, the number of possible choices of indices $h',i,j$ is
bounded by \( N \cdot M \cdot \log N \leq O(N \log^2 N) \).

Consider a specific event \( \mathcal{E}(h, h', i, j) \), and let \( C_{i,j} \subseteq A^{h'}_h \) be the corresponding set of vertices. Then \(|C_{i,j}| = 10M^2\), and each vertex \( v \in C_{i,j} \) is chosen to be in \( U \cup R_{h'} \) with probability at least \( 16 \cdot 2^{d'/M} \), where \( h' \in I_{q'} \). The expected number of vertices of \( C_{i,j} \) added to \( U \) is at least \( 160M \cdot 2^{d'} \), and so from Chernoff bound, the probability that less than \( 80M \cdot 2^{d'} \) vertices are added is bounded by \( e^{-10M} \leq 1/N^{10} \). So the probability that a specific event \( \mathcal{E}(h, h', i, j) \) does not happen is at most \( N^{-10} \), and using the union bound, the probability that any of the events \( \mathcal{E}(h, h', i, j) \) does not happen is at most \( 1/N^2 \). □

**Weight Transfer Procedure** We now focus on the partition \( \{C_{i,j}\}_{i,j} \) induced by \( F_h \) on \( A^{h'}_h \) and define a weight transfer procedure for vertices of \( A^{h'}_h \). For each such \( i : 1 \leq i \leq \log N \), for each \( j : 1 \leq j \leq \alpha_i \), let \( S_{i,j} = \bigcup_{v \in C_{i,j}} D_i(v) \), and \( S'_{i,j} = \bigcup_{v \in C_{i,j}} D_{i+1}(v) \).

Consider a pair of vertices \( v \in C_{i,j}, u \in C_{i,j+1} \). By the definition of the partition \( \{C_{i,j}\} \), \( \tau_i(v) \leq \tau_i(u) \). So if \( v' \in D_i(v) \), then it appears at layer \( L(v') \leq \tau_i(v) \). On the other hand, if \( u' \in D_{i+1}(u) \), then it belongs to layer \( L(u') \geq \tau_i(u) \geq \tau_i(v) \geq L(v') \) (due to the definition of sets \( D_{d'}(u) \)). So we can transfer LP-weight from vertices of \( S_{i,j} \) to vertices of \( S'_{i,j+1} \) without increasing the amortized cost \( \Upsilon_{\tau,h} \) for any value \( \tau \).

We have two types of weight transfer rules. First, for each \( i : 1 \leq i \leq \log N \), for each \( j : 1 \leq j < \alpha_i \), we transfer the LP-weight between \( S_{i,j} \) and \( S'_{i,j+1} \), as follows: if any vertex \( v \in S_{i,j} \) becomes inactive, then half of its current LP-weight \( y^{h'}_h(v) \) is evenly split among all vertices of \( S'_{i,j+1} \). For each \( 1 \leq i \leq \log N \), for each \( 1 \leq j < \alpha_i \), we say that vertices of \( S'_{i,j+1} \) are covered by the first weight transfer rule.

The second weight transfer rule only applies to sets \( C_{1,j} \), where \( 1 \leq j \leq \alpha_1 \). Observe that for each \( v \in C_1 \), \( D_1(v) \) only consists of a single vertex. We are therefore guaranteed that for each \( 1 \leq j < \alpha_1 \), if \( v \in S_{1,j} \) and \( v' \in S_{1,j+1} \) then \( L(v) \leq L(v') \). The second weight transfer rule is that for every \( 1 \leq j < \alpha_1 \), if any vertex \( v \in S_{1,j} \) becomes inactive, then half of its current LP-weight \( y^{h'}_h(v) \) is evenly split among all vertices of \( S_{1,j+1} \). For each
$1 \leq j < \alpha_1$, we say that every vertex in $S_{1,j+1}$ is covered by the second weight transfer rule. This finishes the definition of the weight transfer procedure. Our claim is that every vertex $u \in A_{h'}^h$ is either covered by one of the weight transfer rules, or its ancestor belongs to $R_{h,h'}$. In the former case, we will show that $u$ receives a sufficient contribution to its LP-weight, while in the latter case, $u$ becomes inactive after iteration $h'$.

Claim 27 If $u \in A_{h'}^h$ is not covered by either weight transfer rule, then its ancestor belongs to $R_{h,h'}$.

Proof: Consider some $u \in A_{h'}^h$, and assume that $u \in D_i(v)$ for some $v \in A_{h'}^h$. We consider the following two cases.

First, if $i > 1$, then $v \in C_{i-1}$ because $|D_{i-1}(v)| = 2^{i-2}$ must hold. Assume that $v \in C_{i-1,j}$, for some $1 \leq j \leq \alpha_{i-1}$. If $j > 1$, then $v \in S_{i-1,j}^i$, and it is therefore covered by the first weight transfer rule. If $j = 1$, then $v \in R_{h',h}$.

The second case is when $i = 1$. In this case, either $v \in C_{1,1}$, and then $v \in R_{h',h}$, or else $v \in C_{1,j}$ for some $j > 1$, and so vertex $u$ is covered by the second weight transfer rule. □

It now remains to show that if events $E(h,h',i,j)$ hold, then each vertex $u \in A_{h'}^h$ receives a sufficient contribution. This is done in the following claim.

Claim 28 Let $h' \in I_{q'}$, and assume that at the beginning of iteration $h'$ the LP-weight $y_h^{h'}(u) \geq 2^{q'-1}/M$ for all $u \in A_{h'}^h$. Assume also that events $E(h,h',i,j)$ hold for all $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$. Then the LP-weight of every vertex $u \in A_{h'}^{h+1}$ increases by at least $(2^{q'}/M)^2$ in iteration $h'$, i.e. $y_h^{h+1}(u) \geq y_h^{h'}(u) + (2^{q'}/M)^2$.

Proof: Consider some vertex $u \in A_{h'}^{h+1}$. Since it remains active at the end of iteration $h'$, its ancestor in $F_{h'}$ was not added to $R_{h'}$, so $u$ is covered by one of the weight transfer rules. Assume first that $u$ is covered by the first weight transfer rule, so $u \in S_{i,j+1}^i$, for some $1 < i \leq \log N$, $1 \leq j < \alpha_i$. Since event $E(h,h',i,j)$ holds, at least $80M \cdot 2^{q'}$ vertices of $C_{i,j}$ have been added to $U \cup R_{h'}$ during iteration $h'$. Each such vertex has $2^{i-1}$ descendants in $S_{i,j}$,
and each such descendant has LP-weight of at least $2^{q'-1}/M$. So overall, we have at least 
\[ \frac{1}{2} \cdot 2^{i-1} \cdot 80M \cdot 2^{q'} \cdot \frac{2^{q'-1}}{M} = 2^{i-1} \cdot 20 \cdot 2^{2q'} \] weight that is evenly split among the vertices of $S'_{i,j+1}$.

Recall that $|C_{i,j+1}| \leq 10M^2$, and for each vertex $v \in C_{i,j+1}$, $|D_{i+1}(v)| \leq 2^i$. Therefore, 
\[ |S'_{i,j+1}| \leq 10M^2 \cdot 2^i. \] Each vertex $u \in S'_{i,j+1}$ then receives at least 
\[ \frac{20 \cdot 2^{i-1} \cdot 2^{2q'}}{10M^2 \cdot 2^i} \geq \frac{2^{2q'}}{M^2} \] weight as desired.

Assume now that $u \in S_{1,j+1}$. Again, since event $\mathcal{E}(h, h', 1, j)$ happened, at least $80M \cdot 2^{q'}$ vertices of $C_{1,j}$ have been added to $U \cup R_{h'}$ in iteration $h'$. We then have at least $80M \cdot 2^{q'}$ vertices in $S_{1,j}$ that become inactive, each of which having LP-weight of $2^{q'-1}/M$. So each vertex of $S_{1,j+1}$ receives a contribution of at least 
\[ \frac{1}{2} \cdot 80 \cdot 2^{q'} \cdot 2^{q'-1} \geq 2^{2q'}/M^2. \]

\[ \blacksquare \]

**Corollary 5** Let $h \in I_q$. Assume that events $\mathcal{E}(h, h', i, j)$ happen for all $1 \leq h' < h$, $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$. Then for each phase $q' : 1 \leq q' \leq q$, for each vertex $u \in A_h^{(q')}$ that is active at the beginning of phase $q'$, its LP-weight $y_{h}^{(q')}(u) \geq 2^{q'-1}/M$.

**Proof:** The proof is by induction on $q'$. At the beginning of the first phase, the LP-weight of every vertex $u \in A_h^{(1)}$ for all $h$ is $1/M$, so the claim holds. Assume now that it holds at the beginning of phase $q'$. Let $u \in A_h^{(q'+1)}$ be some vertex that remains active at the end of phase $q'$. We apply Claim 28 for all $h' \in I_{q'}$. This ensures that, for each one of the $M/2^{q'}$ iterations of phase $q'$, the LP-weight of $u$ increases by at least $(2^{q'}/M)^2$, and so the total increase in the LP-weight in phase $q'$ is at least $2^{q'}/M$. \[ \blacksquare \]

We are now ready to complete the proof of Theorem 41. From the above corollary, if events $\mathcal{E}(h, h', i, j)$ happen for all $1 \leq h' < h$, $1 \leq i \leq \log N$, $1 \leq j < \alpha_i$, then at the beginning of iteration $h$, the LP-weight of every vertex in $A_h^h$ is at least $2^{q-1}/M$. Moreover, we have only transferred weight from vertices in $L_{\tau,h}$ to vertices in $L_{\tau',h}$ for $\tau \leq \tau'$. Therefore, throughout the algorithm, the amortized LP-weight $\mathcal{Y}_{\tau,h}$ does not increase for any $\tau$. So at the beginning of iteration $h$, the total LP-weight of active vertices in sets $\mathcal{L}_{1,h}, \ldots, \mathcal{L}_{\tau,h}$ is at least 
\[ \sum_{\tau' \leq \tau} |A_{\tau',h}| \cdot 2^{q-1}/M \] and at most $\mathcal{Y}_{\tau,h}$. Therefore, event $\mathcal{E}_2(h)$ holds with probability at least $1 - 1/N^2$. We have also shown that the sizes of the selected sets $R_{h'}$ are bounded.
by $O(\log^4 N)$.

## 10.2 Directed Layered Graphs

We now describe our LP-rounding algorithm. First, we round each LP-value $x(v)$ up to the next multiple of $1/n$. The resulting fractional solution is clearly still feasible, and its amortized cost increases by at most a factor of 2. Next, we partition the vertices with non-zero LP-weight into $n$ sets $F_1, \ldots, F_n$, as follows. Let $v \in V$ be any vertex with $x(v) = r/n$, where $r > 0$. For each path $P \in \mathcal{P}_v$, let $X(P) = \sum_{v' \in P \setminus \{v\}} x(v')$, and let $h = \min_{P \in \mathcal{P}_v} \{n \cdot X(P)\}$. Then $v$ belongs to sets $F_{h+r'}$ for all $1 \leq r' \leq r$. We view the weight $x(v)$ of $v$ as being evenly distributed among these $r$ sets, so the weight of $v$ w.r.t. $F_{h+r'}$ is $x_{h+r'}(v) = 1/n$, for $1 \leq r' \leq r$. We need the following claim.

**Claim 29** For each $h : 1 \leq h \leq n$, set $F_h$ is a feasible set.

**Proof:** Let $t \in T$ be any terminal and let $P = (s = v_0, v_1, \ldots, v_z = t) \in \mathcal{P}_t$ be any path connecting $s$ to $t$. It is enough to show that for each $h : 1 \leq h \leq n$, $P \cap F_h \neq \emptyset$. For each $\tau : 0 \leq \tau \leq z$, let $Y_\tau = \min_{P \in \mathcal{P}_{v\tau}} \{X(P)\}$. Let $\tau'$ be the largest index for which $Y_{\tau'} < h/n$. We argue that $v_{\tau'} \in F_h$. First notice that $v_{\tau'}$ belongs to $F_{h'+1}, \ldots, F_{h'+r}$ for $h' = nY_{\tau'} < h$ and $r = nx(v_{\tau'})$. There are two cases. If $\tau' = z$, we have that $Y_{\tau'} + x(v_{\tau'}) \geq 1$, and therefore $v_{\tau'} \in F_h$ since $h' < h \leq h' + r$. Otherwise if $\tau' < z$, we have $Y_{\tau'} + x(v_{\tau'}) \geq Y_{\tau' + 1} \geq h/n$, and hence $v_{\tau'} \in F_h$ since, again, $h' < h \leq h' + r$. \qed

Our final solution is the set $F_h^*$ with smallest amortized cost over all sets $F_h$. The next claim shows that this gives an $O(\log n)$-approximation.

**Claim 30** The amortized cost of $F_h^*$ is at most $O(k \log n)$.

**Proof:** We first recall the notation used in the previous sections. We denote, for each $h : 1 \leq h \leq n$ and each $\tau : 1 \leq \tau \leq \lambda$, $X_{\tau,h} = \sum_{v \in F_h \cap L_{\tau}} x_h(v)$ and $\overline{X}_{\tau,h} = \sum_{\tau' \leq \tau} X_{\tau',h}$. For each $h$, let $\tau_h$ denote the layer $\tau : 1 \leq \tau \leq \lambda$ that maximizes $\overline{X}_{\tau,h}/\tau$. 175
For each $h$, we denote the amortized cost of $F_h$ by

$$\text{cost}(F_h) = \max_{\tau \leq \lambda} \left\{ \frac{1}{\tau} \left( \sum_{\tau' \leq \tau} |F_h \cap L_{\tau'}| \right) \right\}$$

or equivalently,

$$\text{cost}(F_h) = \max_{\tau} \left\{ \frac{1}{\tau} \sum_{\tau' \leq \tau} X_{\tau',h} \cdot n \right\} = \frac{n}{\tau_h} \cdot X_{\tau_h,h},$$

since the LP-weight $x_h(v)$ of every vertex $v \in F_h$ w.r.t. $F_h$ is $1/n$. We now have that:

$$\text{cost}(F_{h^*}) \leq \frac{1}{n} \sum_{h=1}^{n} \text{cost}(F_h) = \sum_{h=1}^{n} X_{\tau_h,h}/\tau_h$$

Partition the indices $h : 1 \leq h \leq n$ into $\lceil \log \lambda \rceil$ subsets $I_1, \ldots, I_{\lceil \log \lambda \rceil}$, as follows: $I_q = \{h | 2^{q-1} \leq \tau_h < 2^q\}$, for $1 \leq q \leq \lceil \log \lambda \rceil$. For each such set $I_q$, we bound the summation for indices $h \in I_q$ as follows:

$$\sum_{h \in I_q} X_{\tau_h,h}/\tau_h \leq \sum_{h \in I_q} X_{2^q,h}/2^{q-1}$$

$$\leq 2 \sum_{h=1}^{n} X_{2^q,h}/2^q$$

$$\leq 2X_{2^q}/2^q \leq 4k$$

(the additional factor of 2 comes from the original rounding to produce a $1/n$-integral solution). Summing up over all sets $I_q$, we get that $\text{cost}(F_{h^*}) \leq O(\text{OPT} \cdot \log \lambda) \leq O(\log n)\text{OPT}$. \hfill \Box

### 10.3 General Graphs

In this section, we give a $O(n^{1/3} \log n)$-approximation algorithm for directed graphs. We first state the result for directed layered graph from the previous section which will be used iteratively as a subroutine.
Theorem 43 Suppose we are given a directed layered graph $G = (V', E')$, source $s$ and terminal set $T$ as an input. Given a feasible solution $z$ to (LP-2) of cost $k'$, we can find, in polynomial time, a feasible solution $S' = \{U_\tau\}$ of cost at most $O(k' \log n)$.

Now we are ready to describe our algorithm. Suppose we are given a directed graph $G = (V, E)$ with a source $s$ and terminal set $T$. Let $\{x^*_v\}$ be an optimal solution to (LP-1) of cost $k^* \geq 1$, and for convenience, we let $\gamma = \left\lceil n^{1/3} \right\rceil$. Our algorithm has two steps. In the first step, we produce a set $S' = \{U'_\tau\}_{\tau=1}^n$ that ensures that all the paths of length at most $\gamma$ are not fire spreading paths. Then in the second step, we create another set $S''$, guaranteeing that the remaining paths are not fire spreading.

First Step: Let $G_1 = G$. This step has $\gamma$ iterations. In iteration $i$, we create a directed layered graph $H \subseteq G_i$ where vertices $V(H)$ are defined by a layering $V(H) = \bigcup_\tau \mathcal{L}'_\tau$. Layer $\mathcal{L}'_j$ contains all vertices in $G_i$ of distance $j$ from $s$. We define the edges $E(H)$ to contain the edges of $G_i$ that are directed from layer $\mathcal{L}'_j$ to layer $\mathcal{L}'_{j+1}$ for some $j$. In other words, graph $H$ is a subgraph of $G_i$ containing only edges that lie on some shortest $s$-$t$ paths for some terminal $t \in T$. Graph $H$ is associated with the LP solution $\{z_v\}$ where each vertex $v$ at layer $\mathcal{L}'_j$ of $H$ has LP-value $z_v = \sum_{\tau \leq j} x^*_v$.

Claim 31 $\{z_v\}_{v \in V(H)}$ is a feasible solution for (LP-2), and the cost of $z$ is at most $k^*$.

Proof: First we show that the first set of constraints are satisfied. We now fix some time step $\tau$ and bound the amortized cost at layer $\tau$. It is clear that $\sum_{\tau' \leq \tau} \sum_{v \in \mathcal{L}'_{\tau'}} z_v = \sum_{\tau' \leq \tau} \sum_{v \in \mathcal{L}'_{\tau'}} \sum_{\tau'' \leq \tau'} x^*_{v''}$, which is at most $\sum_{v \in V} \sum_{\tau'' \leq \tau} x^*_{v''} \leq k^* \tau$. Next consider a terminal $t$ and any path $P$ from $s$ to $t$ in $H$. Since path $P = (s = v_0, v_1, \ldots, v_\rho = t)$ exists in graph $G_i$, it must satisfy the second set of constraints in (LP-1). That is, $\sum_{i=1}^\rho z_{v_i} = \sum_{i=1}^n \sum_{\tau \leq i} x^*_{v_i} \geq 1$. 

We then apply Theorem 43 to obtain a set $\tilde{S}_i = \{\tilde{U}_\tau^i\}_{\tau=1}^n$ where each set $\tilde{U}_\tau^i$ contains at most $O(k^* \log n)$ vertices. Next we remove vertices in $\tilde{S}_i$ (and all of their adjacent edges).
from $G_i$ to obtain the graph $G_{i+1}$, and proceed to iteration $i + 1$. This completes the description of iteration $i$.

After $\gamma$ iterations, we combine the solution over iterations to obtain $S' = \{U'_\tau\}_{\tau=1}^n$ by $U'_\tau = \bigcup_{i=1}^{\gamma} \tilde{U}_i^\tau$. The cost of $S'$ is at most $O(k^*\gamma \log n)$. Let $G' = G_{\gamma+1}$ be the output from the last iteration, which will be the input of the second step. We prove the following two crucial properties.

**Lemma 25** Let $t$ be any terminal in $G$ and $P$ be an $s$-$t$ path (in original input graph $G$) that contains some vertices in $S'$. Then $P$ is not a fire spreading path with respect to $S'$.

**Proof:** We write path $P$ as $P = (s = v_1, \ldots, v_z = t)$. Let $i$ be the minimum integer such that $P \cap \tilde{S}_i \neq \emptyset$, so $P$ is a path in $G_i$ but not in $G_{i+1}$. Now we fix iteration $i$ and consider a subgraph $H$ created in this iteration. We argue that $P$ is protected by set $S_i$. Let $v_j$ be the vertex on $P$ such that the distance from $s$ to $v_j$ in $H$ is minimum. So $v_j$ belongs to some layer $L_{j'}$ for some $j' \leq j$. Therefore, $v_j \in \tilde{U}_{j''}^i$, for some $j'' \leq j'$, which implies that path $P$ is not fire spreading with respect to $S'$.

**Lemma 26** In the beginning of iteration $i$, for all terminal $t \in T$, there is no path of length less than $i$ from $s$ to $t$ in $G_i$.

**Proof:** We prove this claim by induction on $i$. For $i = 1$, the lemma follows trivially. Now assume that in the beginning of iteration $i$ there is no path of length at most $i - 1$, and consider the execution of iteration $i$. Let $t$ be any terminal and $P$ be any $s$-$t$ path of length $i$ in $G_i$. Since there is no path of length $i - 1$, $P$ must be a shortest $s$-$t$ path, which implies that the $P$ is a path in graph $H$. So path $P$ must contain some vertex in $\bigcup_{\tau} \tilde{U}_i^\tau$. Removing vertices in $\tilde{S}_i$ therefore eliminates all such paths $P$.

We note that, as a corollary of the above lemma, there is no path of length at most $\gamma$ from source $s$ to any terminal in $G'$. Now let $\mathcal{P}$ be the set of all paths from source $s$ to some terminal $t$ in $G$. Define $\mathcal{P}'$ to contain all paths that contain some vertices saved by $S'$, so from the lemma, we know that paths in $\mathcal{P}'$ are not fire spreading. Moreover, paths in $\mathcal{P}''$
must appear in $G'$, and the goal of our second step is to ensure that these paths are not fire spreading.

**Second Step:** We consider the BFS layering of $G'$ into $\bigcup_{j=1}^{n} L_j$ where $L_j$ contains vertices in $G'$ of distance $j$ from the source $s$. We claim that there is a set $L_j$ that has low “amortized cost”.

**Claim 32** There is a layer $L_j$, for $j \leq \gamma$, such that $|L_j| \leq 2j\gamma$.

**Proof:** Assume otherwise that for each $j = 1, \ldots, \gamma$, we have $|L_j| > 2j\gamma$. Summing over all such $j$, we get $|V(G')| \geq 2\gamma \cdot \sum_{j=1}^{\gamma} j > \gamma^3 \geq n$, a contradiction. □

Let $L_{j^*}$ be such a layer. Clearly, $L_{j^*}$ is a vertex cut in graph $G'$ separating $s$ from terminals $T$, because in graph $G'$ each terminal in $T$ must be of distance at least $\gamma$ from $s$. We then create a set $S'' = \{U''_{\tau}\}_{\tau=1}^{j^*}$ where vertices in $L_{j^*}$ are divided equally into $j^*$ sets $U''_{\tau}$ for $\tau = 1, \ldots, j^*$. Therefore, $|U''_{\tau}| \leq 2\gamma$. We construct the solution $S = \{U_{\tau}\}_{\tau=1}^{n}$ by combining $S'$ with $S''$, i.e. $U_{\tau} = U'_{\tau} \cup U''_{\tau}$ for all $\tau$. The cost of $S$ is at most $O(k^*\gamma \log n)$.

It is now clear that path $P \in P''$ is not fire spreading with respect to $S''$. Therefore, $S$ is a feasible solution, and this concludes the proof.
CHAPTER 11
LOWER BOUNDS ON THE LP INTEGRALITY GAP

In this chapter, we discuss the lower bound on the integrality gap of (LP-1), (LP-2) and (LP-3) for general graphs, directed layered graphs, and trees. For trees, we show that our integrality gap result holds even for apparently stronger LP relaxation. We present the result for trees in Section 11.1, directed layered graphs in Section 11.2, and general graphs in Section 11.3.

11.1 Trees

In this section we present a lower bound of $\Omega(\log^* n)$ on the integrality gap of (LP-3), matching the upper bound to within a constant factor. Recall that, for the tree case, we may assume that any optimal solution $U$ satisfies $|U \cap L_\tau| \leq k$ for all $\tau$.

11.1.1 Stronger LP Relaxation

There is a natural way to strengthen (LP-3) by incorporating a simple observation as follows. Let $G = (V, E)$ be our input instance. For each vertex $v$, denote by $G_v$ a subtree of $G$ rooted at $v$. Suppose we know the value $k$ of the optimal solution’s cost $S$. If we solve (LP-3) on the subgraph $G_v$ and it turns out that the solution cost is more than $k$, it means that vertex $v$ must not burn w.r.t. solution $S$, so we can simply replace the whole subtree $G_v$ by a terminal without increasing the cost of the optimal solution. This operation is summarized formally below.

$\text{ProbeAndTrim}(G, T, k)$

- For each $v \in V(G)$,
  - Solve (LP-3) on the subgraph $G_v$. 

180
- If the optimal value returned by the previous step is more than \(k\), we remove from \(G\) all vertices of \(G_v\) except for \(v\), and set the new set of terminals to be \((T \setminus T_v) \cup \{v\}\).

- Let \((G', T')\) be the tree obtained after all vertices of \(G\) have been processed.

The following claim is immediate.

**Claim 33** Let \(S\) be a feasible integral solution of cost \(k\) and \(G'\) be the new graph obtained by applying \(\text{PROBEANDTRIM}(G, v, k)\) for some \(v \in V(G)\). Then \(S\) is feasible for \(G'\).

We say that an instance is \(k\)-infeasible if repeatedly applying the \(\text{PROBEANDTRIM}\) operations results in a single vertex \(s\), which is a terminal itself. Otherwise, we say that the instance is \(k\)-feasible. In the following proposition, we argue that this procedure gives us a “relaxation” that is at least as good as (LP-3).

**Proposition 1** Let \(k^*\) be the optimal LP cost obtained from solving (LP-3) on graph \(G\), and \(\tilde{k}\) be the minimum \(k\) such that the instance is \(k\)-feasible. Then

\[k^* \leq \tilde{k} \leq \text{OPT}\]

**Proof:** It is clear that \(k^*, k \leq \text{OPT}\). We only need to argue that \(k^* \leq \tilde{k}\). Assume that the instance is \(\tilde{k}\) feasible, so we have an associated LP solution \(\{x(v)\}_{v \in \tilde{G}}\), where \((\tilde{G}, \tilde{T})\) is the final instance to which applying the operation \(\text{PROBEANDTRIM}\) does not change the result. We define a solution \(x'\) where \(x'(v) = x(v)\) for \(v \in V(\tilde{G})\) and \(x'(v) = 0\) otherwise.

The terminals in \(\tilde{T}\) are either the original terminals in \(T\) or the new terminals created along the process. We write \(\tilde{T} = \tilde{T}_1 \cup \tilde{T}_2\), where \(\tilde{T}_1 \subseteq T\) and for each terminal \(t \in \tilde{T}_2\), there is a corresponding vertex \(\varphi(t) \in V(G)\).

Now consider a terminal \(t \in T\). If \(t \in \tilde{T}_1\), it must be protected by solution \(\{x(v)\}\). Otherwise, terminal \(t\) is in a subtree \(G_{\varphi(t')}\) for some \(t' \in \tilde{T}_2\), and the feasibility of LP solution guarantees that \(\sum_{v \in P_{t'}} x(v) \geq 1\). \(\square\)
11.1.2 Construction

We use the following two simple spider graphs. A type-1 spider of width $w$ and height $h$ consists of $w$ paths of length $h$ each. The paths are completely disjoint, except that they all share one common endpoint called the *spider head*. The other endpoints of the $w$ paths are called the *spider feet*. Type-2 spider of width $w$ and height $h$ is defined similarly, except...
that the lengths of the $w$ paths vary, with the $i$th path, for $1 \leq i \leq w$, having length $h + i$ (see Figures 11.1 and 11.2). We view each such spider as a tree rooted at its head.

The integrality gap instance $G$ uses a parameter $M$ which will be an intended integrality gap lower bound. The instance consists of $M + 1$ levels. Each level $i : 0 \leq i \leq M$ is a forest. The set of roots of its trees is denoted by $A_i$, and the set of their leaves by $B_i$. All vertices of $A_i$ lie in the same layer of $G$, and the same is true for $B_i$. For convenience, when defining level $i$, we renumber its layers, so $A_i$ is layer 0 of level $i$ and $B_i$ is the last layer, whose index is denoted by $\lambda_i$. Vertices in set $B_i$ are partitioned into $\ell_i$ subsets, $B_i = \bigcup_{j=1}^{\ell_i} B_i^j$. We now proceed to describe the levels. Level 0 consists of only two layers, $A_0 = \{s\}$ and $B_0 = \{s_1, \ldots, s_{2M}\}$ with edges connecting $s$ to every vertex of $B_0$. Set $B_0$ is partitioned into $\ell_0 = 2M$ subsets $B_0^1, \ldots, B_0^{2M}$, with $B_0^j = \{s_j\}$ for $1 \leq j \leq 2M$. We now describe level $i$ for some $i > 0$. Consider the last layer $B_{i-1}$ of level $i-1$ and its corresponding partition $\{B_{i-1}^j\}_{j=1}^{\ell_{i-1}}$. Let $I$ be the set of all ordered $M$-tuples of vertices in $B_{i-1}$, where the vertices in each $M$-tuple belong to distinct sets of the partition. Therefore, $I$ consists of all ordered $M$-tuples $(v_1, \ldots, v_M)$, such that, for each $1 \leq r \leq M$, if $v_r \in B_{i-1}^{j_r}$, then all indices $j_1, \ldots, j_M$ are distinct. Let $\ell_i = |I| \leq |B_{i-1}|^M$. We add a set $A_i^\rho = \{v_1^\rho, \ldots, v_M^\rho\}$ of vertices for each $\rho \in I$ and set $A_i = \bigcup_{\rho \in I} A_i^\rho$. For each $\rho = (v_1, \ldots, v_M) \in I$, for each $r : 1 \leq r \leq M$, there is an edge $(v_r, v_{\rho}^M)$, where $v_r \in B_{i-1}$. This finishes the definition of set $A_i$. Level $i$ contains, for each $M$-tuple $\rho = (v_1^\rho, \ldots, v_M^\rho) \in I$, a gadget $H_i^{\rho_1}$, consisting of $M$ identical trees, whose roots are the $M$ vertices of $A_i^\rho$. The set of the leaves of the $M$ trees, lying in layer $\lambda_i$, is denoted by $B_i^\rho$. We then set $B_i = \bigcup_{\rho \in I} B_i^\rho$, with the corresponding partition of $B_i$ into $\ell_i$ subsets $\{B_i^\rho\}_{\rho \in I}$. We set $\lambda_i = \sum_{j=1}^{\ell_i} (2M)^j$, and we assume that we are given some arbitrary ordering $\rho_1, \ldots, \rho_{\ell_i}$ of the tuples in $I$.

We now fix an $M$-tuple $\rho_j = (v_1, \ldots, v_M) \in I$ and define the corresponding gadget $H_i^{\rho_j}$. The gadget consists of $M$ copies of tree $T_j$, rooted at vertices $v_1^{\rho_j}, \ldots, v_M^{\rho_j}$. Let $v$ denote the root of $T_j$, and let $b_j = \sum_{1 \leq j' < j} (2M)^{j'}$. We add a type-2 spider of width $(2M)^j$ and height $b_j$, whose head is $v$. The feet of the spider are called special vertices. If the current level
i \neq M$, then we perform the following additional step. For each special vertex $u$ of $T_j$, if $u$ lies at layer $b_j + h$, for $1 \leq h \leq (2M)^j$, then we add a type-1 spider of width $2M\lambda_i$ and height $\lambda_i - h - b_j$ whose head is $u$. This ensures that all leaves of tree $T_j$ lie in layer $\lambda_i$. The final gadget $H_{i}^{\rho_j}$ is obtained by attaching a copy of $T_j$ as a subtree to each vertex in $A_{i}^{\rho_j}$. This finishes the description of level $i$. Notice that there are at most $M$ special vertices at each layer of level $i$. The set $T$ of terminals is the set of all special vertices lying at the last level $M$. Let $N$ denote the construction size.

11.1.3 Analysis

We show that instance $G$ is 1-feasible, and in particular the cost of the fractional solution is 1. On the other hand, we prove that the cost of any feasible integral solution is at least $M$. We start by bounding $N$, the construction size.

Since $N \leq O(|B_M|^2)$, it is enough to bound the sizes of sets $B_i$ for all $i$. Recall that $|B_0| = 2M$. For $i \geq 1$, we have that $\ell_i \leq |B_{i-1}|^M$. Therefore, $\lambda_i \leq (2M)^{\ell_i+1} \leq (2M)^{|B_{i-1}|^{M+1}}$. The number of special vertices at level $i$ is at most $M\lambda_i$, and each special vertex has $2M\lambda_i$ descendants in set $B_i$, so overall $|B_i| \leq 2M^2\lambda_i^2 \leq 2M^2 \cdot (2M)^2 |B_{i-1}|^{M+2} \leq (2M)^{|B_{i-1}|^{2M}}$. For simplicity, denote $Y_i = |B_i|$ for all $i$, and $m = 2M$. We then have the following recurrence: $Y_0 = m$, and for $0 \leq i < M$, $Y_{i+1} \leq m^{Y_i}$. We use the following claim.

Claim 34 For $0 \leq i \leq M$, $\log^{(i)} N \leq m^2 Y_{M-i}^m$.

Proof: The claim holds trivially for $i = 0$. Assume that the claim holds for some $i$. Then $\log^{(i+1)} N \leq \log(m^2 Y_{M-i}^m) \leq 2 \log m + m \log Y_{M-i}$. Replacing $Y_{M-i} \leq m^{Y_{M-i-1}}$, we get that:

$$\log^{(i+1)} N \leq 2 \log m + m \log \left( m^{Y_{M-i-1}} \right) \leq 2 \log m + m \cdot Y_{M-i-1} \log m \leq m^2 Y_{M-i-1}^m$$
for a large enough $m$.

Applying the claim for $i = M$, we get that $\log^2 N \leq m^2 y_0^m \leq (2M)^{O(M)}$. Clearly, taking the logarithm for $O(\log^{2} M) \leq M$ more steps will get the number on the right hand side below 1. Therefore, $\log^2 (2M) N \leq 1$, and $M = \Omega(\log N)$.

We now proceed to analyze the fractional solution cost.

**Lemma 27** *(fractional solution’s cost)* There is a fractional solution of cost 1 for instance $G$, and moreover $G$ is 1-feasible.

**Proof:** The fractional solution simply assigns a $1/M$ value to each special vertex. Since every layer contains at most $M$ special vertices, the cost of the solution is 1. It is also easy to see that for every level $i$, if $u \in B_i$ is a descendant of $v \in A_i$, then the path connecting $v$ to $u$ contains exactly one special vertex. Therefore, there are $M$ special vertices on every root-to-terminal path, and the solution is feasible.

We now show that instance $G$ is 1-feasible. Consider some non-terminal vertex $v$, its subtree $G_v$ and the corresponding subset $T_v$ of terminals. If $v$ has only one child, then there is a trivial fractional solution of cost 1 to instance $(G_v, T_v)$, in which an LP-value of 1 is assigned to the child of $v$. Assume now that $v$ has more than 1 child. This can only happen if $v$ is a special vertex, or $v \in A_i$, or $v \in B_i$ for some level $i$.

Assume first that $v \in A_i$ for some level $i : 1 \leq i \leq M$ (the case where $i = 0$, and $v = s$ has been analyzed above). Consider the sub-tree $G_v^i$ rooted at $v$, restricted to only vertices of level $i$. This subtree contains at most one special vertex at each layer. Therefore, there is a feasible solution of cost 1 to instance $(G_v, T_v)$, where LP-weight of 1 is assigned to every special vertex of $G_v^i$.

Assume now that $v \in B_{i-1}$ for some $i : 1 \leq i \leq M$. Let $v^{\rho_1}, \ldots, v^{\rho_h}$ be the children of $v$. Due to the definition of $I$, each vertex $v^{\rho_j} : 1 \leq j \leq h$ participates in a distinct gadget $H_{i_j}^b$ at level $i$. This again ensures that all special vertices lying in the sub-tree of $v$ at level $i$ belong to distinct layers. A feasible solution of cost 1 is then obtained by placing an LP-weight of
1 on each such special vertex.

Finally, assume that \( v \) is a non-terminal special vertex at level \( i : 1 \leq i < M \). Due to the definition of the partition of \( B_i \), all descendants of \( v \) in \( B_i \) belong to the same set \( B^j_i \) of the partition. This ensures that all descendants of \( v \) in \( A_{i+1} \) participate in distinct gadgets \( H^\rho_{i+1} \), and therefore the special vertices in the sub-tree of \( v \) lying at level \( i + 1 \) all belong to distinct layers. Again, a feasible fractional solution of cost 1 for the corresponding instance is obtained by assigning an LP-value 1 to all special vertices lying at level \( i + 1 \).

It now only remains to show that the optimal integral solution cost is at least \( M \). Let \( \lambda \) be the total number of layers in our construction, and let \( S = \{U_\tau\}_{\tau \geq 1} \) be any solution of cost at most \( M - 1 \). From Claim 21, we can assume, w.l.o.g., that for all \( 1 \leq \tau \leq \lambda \), \( U_\tau \subseteq L_\tau \), where \( L_\tau \) is layer \( \tau \) of the tree, and \( U_\tau = \emptyset \) for \( \tau > \lambda \). We assume that we are given a solution \( S = \{U_\tau\}_{\tau = 1}^\lambda \) of this form. To simplify the analysis of this part, we view the set of vertices \( B_0 = \{s_1, \ldots, s_{2M}\} \) as special vertices, and we define 2\( M \) gadgets \( H^\rho_0 : 1 \leq \rho \leq 2M \) at level 0, where gadget \( H^\rho_0 \) consists of a single vertex \( s_\rho \). In order to prove that \( S \) is not a feasible solution, it is enough to show that at least one terminal is not protected. The next lemma will then complete the analysis of the integrality gap.

**Lemma 28** Let \( S = \{U_\tau\}_{\tau = 1}^\lambda \) be any integral solution of cost at most \( M - 1 \). Then for every \( i : 0 \leq i \leq M \), there is a set \( V_i \) of \( M + 1 \) special vertices lying at level \( i \), that are not protected by \( S \). Moreover, all vertices in \( V_i \) belong to distinct level-\( i \) gadgets \( H^\rho_i \).

**Proof:** The proof is by induction. Consider first \( i = 0 \). Since \( B_0 \) contains \( 2M \) vertices, and the solution is only allowed to save \( M - 1 \) of them, at least \( M + 1 \) vertices in \( B_0 \) are not protected, and they belong to distinct gadgets by definition. Assume now that the lemma is true for some \( 0 \leq i < M \), and let \( V_i = \{u_1, \ldots, u_{M+1}\} \) be the corresponding set of unprotected special vertices at level \( i \). We need the following claim:

**Claim 35** Let \( u \) be any special vertex lying at level \( i \) that is not protected by \( S \). Then at least one descendant of \( u \) in \( B_i \) is not protected by \( S \).
Proof: Assume otherwise. Recall that there is a type-1 spider of width $2M\lambda_i$ and height at most $\lambda_i$ rooted at $u$. Let $Q$ be the set of the leaves of this spider, $Q \subseteq B_i$. Protecting each vertex in $Q$ requires saving a distinct vertex of the spider. As vertex $u$ is not protected, solution $S$ can only protect at most $(M - 1)\lambda_i$ vertices of the spider, and hence protect at most $(M - 1)\lambda_i$ vertices of $Q$.

For each vertex $u_j \in V_i$, let $w_j$ be any descendant of $u_j$ in $B_i$ that is not protected by $S$. Since vertices $\{u_j\}_{j=1}^{M+1}$ belong to distinct level-$i$ gadgets, vertices $w_1, \ldots, w_{M+1}$ belong to distinct sets in the partition of $B_i$. We say that an $M$-tuple $\rho \in I$ is bad iff $\rho \subseteq \{w_1, \ldots w_{M+1}\}$. Clearly, there are at least $(M + 1)! \geq 2M$ bad ordered tuples $\rho \in I$. Given a bad tuple $\rho \in I$, we say that the corresponding level-$(i + 1)$ gadget $H_{i+1}^\rho$ is bad iff none of the vertices in set $A_{i+1}^\rho$ is protected by $S$. Since we have at least $2M$ bad tuples $\rho$, there are $2M$ corresponding disjoint sets $A_{i+1}^\rho$ of vertices in $A_{i+1}$, while $S$ can only save $M - 1$ vertices of $A_{i+1}$. Therefore, there are at least $M + 1$ bad level-$(i + 1)$ gadgets. It now only remains to prove that each such bad gadget contains at least one special vertex that is not protected by $S$.

Claim 36 Let $H = H_{i+1}^\rho$ be a bad level-$(i + 1)$ gadget. Then at least one special vertex of $H$ is not protected by $S$.

Proof: Assume that $\rho$ is the $j$th tuple in the corresponding set $I$. Consider the set $A_{i+1}^\rho$ of vertices. Each vertex $v \in A_{i+1}^\rho$ is a head of a type-2 spider of width $(2M)^j$ and height $b_j$, and none of the vertices in $A_{i+1}^\rho$ is protected by $S$. Therefore, in order to protect all special vertices of these spiders, $S$ has to save at least $M \cdot (2M)^j$ vertices in layers $1, \ldots, b_j + (2M)^j$ of level $(i + 1)$. Since at most $(M - 1)$ vertices can be saved in each layer, it is enough to show that $M \cdot (2M)^j > (M - 1)(b_j + (2M)^j)$, or equivalently that $(2M)^j > (M - 1)b_j$. Indeed, $b_j = \sum_{j'=1}^{j-1}(2M)^{j'} \leq \frac{(2M)^j}{2M-1} < \frac{(2M)^j}{M-1}$. \hfill \Box
11.2 Directed Layered Graphs

We sketch the lower bound of $\Omega(\log n)$ on the integrality gap of the LP for RMFC on directed layered graphs due to Khanna and Olver [67]. Let $d$ be a parameter, and $\lambda = \lceil e^d \rceil$. The construction has $\lambda + 1$ layers $L_0, \ldots, L_\lambda$ where $L_0 = \{s\}$, and $L_\tau$ contains $\tau d$ vertices for all $\tau \geq 1$. The terminal set is $T = L_\lambda$. For each $0 \leq \tau \leq \lambda - 1$, there is an edge $(u, v)$ for every $u \in L_\tau, v \in L_{\tau+1}$.

Consider the following fractional solution: for each $v \in L_\tau$, we set $x(v) = 1/(\tau d)$. It is easy to see that this is a feasible fractional solution: for each terminal $t \in T$, for each path $P \in \mathcal{P}_t$, we have $\sum_{u \in P} x(u) = \frac{1}{d} \sum_{\tau \leq \lambda} \frac{1}{\tau} \geq \frac{1}{d} \ln \lambda \geq 1$. The total LP-weight of each layer is exactly 1, so the cost of the solution is 1. It is easy to see that the cost of the optimal integral solution for this instance is $d$. Since the construction size $n = d(\sum_{\tau \leq \lambda} \tau) = O(d\lambda^2) = O(de^{2d})$, the integrality gap is $d = \Omega(\log n)$.

11.3 General Graphs

In this section, we show an integrality gap example of (LP-1) which gives a lower bound of $\Omega(n^{1/6})$. We will be using parameter $\alpha$ to denote the intended integrality gap value, and $\ell = 4\alpha^2$.

The Construction: We start by constructing a layered graph containing $\alpha k + 1$ layers, $\bigcup_{j=0}^{\alpha \ell} \mathcal{L}_j$ where each layer $\mathcal{L}_j$ contains $N = 10\alpha \ell$ vertices. Between any consecutive layers $\mathcal{L}_j$ and $\mathcal{L}_{j+1}$, we have a directed edge $(u, v)$ between every pair of vertices $u \in \mathcal{L}_j$ and $v \in \mathcal{L}_{j+1}$. Then we connect the source $s$ to every vertex $v$ in the first layer $\mathcal{L}_0$ by a directed edge $(s, v)$. Let $T$ be a set of terminals, $|T| = \alpha \ell/2$. We connect an edge $(u, t)$ for every vertex $u \in \mathcal{L}_{\alpha \ell}$ and every vertex $t \in T$. These layers $\mathcal{L}_j$ are also called fat layers. Layer $\mathcal{L}_j$ where $j$ is a multiplicity of $\alpha$ is denoted as special layer, so there are $\ell + 1$ special layers: $\mathcal{L}_0, \mathcal{L}_\alpha, \ldots, \mathcal{L}_{\ell \alpha}$. 

188
Next we construct $\ell$ thin layers $L'_1, \ldots, L'_\ell$ where each thin layer contains $\ell$ vertices. For each $i = 1, \ldots, \ell$, we perform the following: Let $L_{\alpha(i-1)}$ and $L_{\alpha i}$ be consecutive special layers. We add directed edges $(u, v)$ and $(v, w)$ between every choice of $u \in L_{\alpha(i-1)}, v \in L'_i, w \in L_{\alpha i}$.

This concludes the construction’s description. We show below that the cost of the fractional solution is 3 while the integral solution costs at least $\Theta(\alpha)$. Therefore, this gives the integrality gap of $\Omega(\alpha)$. The number of vertices in the graph is $|V| = \Theta(N\alpha\ell) = \Theta(\alpha^6)$, so we get the integrality gap of $\Omega(\alpha) = \Omega(|V|^{1/6})$.

**Fractional solution:** We define the fractional solution $\{x^\tau_v\}$ as follows. For each $i = 1, \ldots, \ell$, for each $v \in L'_i$, we let $x^i_v = 2/\ell$. Moreover, we write terminal set as $T = \{t_1, \ldots, t_{|T|}\}$, and for each time step $\tau$, we set the LP-value $x^\tau_{i_r} = 1$. In other words, in time step $i$, the solution saves (i) $2/\ell$ fraction of each vertex in the $i$th thin layer, and (ii) one terminal vertex. So the cost of the solution is 3. We argue below that the solution is feasible.

Let $t$ be any terminal in $T$. Let $P = (v_0, v_1, \ldots, v_z = t)$ be a path from $s$ to $t$. We show that the total LP-weight on $P$ is at least one. We consider two cases. First if path $P$ visits thin layers at least $\ell/2$ times, and let $v_{i_1}, \ldots, v_{i_{\ell/2}}$ be the vertices on $P$ that lie on thin layers $L'_{j_1}, \ldots, L'_{j_{\ell/2}}$ respectively. Observe that $j_r \leq i_r$ for all $r : 1 \leq r \leq \ell/2$, and this would ensure that the total LP-weight $\sum_{r=1}^{\ell/2} x^\tau_{v_{i_r}}$ is at least $2/\ell$. Hence, the total LP-weight on path $P$ is at least one. Next consider the case when $P$ visits the thin layers less than $\ell/2$ times. In this case the length of path $P$ must be at least $z \geq \alpha \ell/2$. Therefore, terminal $t$ would have been already saved at the time step $z$.

To sum up the ideas, the fire that spreads through the long paths will reach the terminal after all terminals are saved, while the fire spreading along the short paths must visit the thin layers often enough to receive the total LP-weight of at least one.
Integral solution: First we notice that, in the construction, the longest $s$-$t$ path has length $\ell \leq 2\alpha \ell$. Let $S = \{U_\tau\}^\ell_{\tau=1}$ be any integral solution of cost at most $\alpha/10$. We show that this solution cannot be feasible. Consider the solution in the first $4\ell$ steps: $U_1, \ldots, U_{4\ell}$ and $S' = \bigcup_{\tau=1}^{4\ell} U_\tau$. Since $S'$ saves at most $\alpha/10$ vertices per time step, $|S'| \leq 2\alpha \ell / 5 < |T|$, so there is a terminal $t \in T$ that is not saved directly by $S'$. We show that terminal $t$ burns in step $4\ell$ by identifying the fire-spreading path from $s$ to $t$ of length at most $4\ell$.

We say that a thin layer $L_j'$ is covered by $S'$ if it is completely contained in $S'$. We first note the following two observations.

Observation 5 The number of thin layers that can be covered by $S'$ is at most $|S'|/\ell \leq \alpha/2$.

Observation 6 For each layer $L_i$, there is a vertex $v \in L_i$ such that $v \notin S'$.

For each $i : 0 \leq i \leq \alpha k + 1$, we let $v_i \in S'$ be such a vertex in $L_i$. For each $j : 1 \leq j \leq \ell$ such that $L_j'$ is not covered by $S'$, we let $u_j$ be a vertex in $L_j'$ that does not belong to $S'$. Now we construct a fire-spreading path with respect to $S'$ of length at most $4k$. We define sub-paths $P_j$ and concatenate them to obtain our final path $P$.

For each $j$ such that $L_j'$ is covered by $S'$, we define the sub-path

$$P_j = (v_{\alpha(j-1)}, v_{\alpha(j-1)+1}, \ldots, v_{\alpha j})$$

That is, the sub-path goes along the fat layers. For each $j$ such that $L_j'$ is not covered by $S'$, we define $P_j = (v_{\alpha(j-1)}, u_j, v_{\alpha j})$. We then combine the paths $P_1, \ldots, P_\ell$ by unifying the common endpoints appropriately, and then add the source and sink to the beginning and the end of the combined path, to get path $P$. The length of each long sub-path is $\alpha$ while the rest has length 2 each. Hence, the total length of $P$ is at most $\alpha^2/2 + 2\ell < 4\ell$. 

190
CHAPTER 12
CONCLUSION AND OPEN PROBLEMS

We have shown tight bounds on the integrality gap of natural LP relaxations of the RMFC problem for trees and directed layered graphs. For general graphs, a gap remains between the upper bound and lower bound, i.e. we have a $\tilde{O}(n^{1/3})$ upper bound v.s. $\Omega(n^{1/6})$ lower bound.

The main result in this part is an $O(\log^* n)$ approximation algorithm for RMFC on trees. To prove this result, we have introduced what we call “weight transfer scheme” that allows us to transfer the “unused” LP-weight from one vertex to another. We believe that this technique itself will find its further applications, especially in covering problems where the underlying set system has certain structures.

We conclude this part by listing some open problems.

1. The approximability of RMFC problems is still poorly understood. In general graphs, it is known to be hard to approximate to within a factor better than 2, while the best approximation algorithm is still $\tilde{O}(n^{1/3})$. It would be interesting to try to narrow down this gap.

2. It is interesting to study RMFC on other natural and special graph classes, such as planar graphs, bounded treewidth graphs, and random graphs. To the best of our knowledge, no approximation algorithm bounds are known for such graph classes.

3. In directed layered graphs, it seems that the natural LP relaxation is not strong enough, as the $\Omega(\log n)$ integrality gap example does not seem to capture the “hardness” of the problem at all. There are certain sets of constraints that hold in the integral world and therefore can be added to the LP. It is an interesting problem to come up with natural constraints that can be added to (LP-2) to narrow down the integrality gap. We believe that getting a sub-logarithmic approximation for this special case along this line should be possible.
4. A closely related problem is the Firefighters problem, in which our goal is to protect as many vertices in the terminals as possible. This problem is known to be \(n^{1-\epsilon}\) hard to approximate in general graphs [9], but the Firefighters problem on the trees admits a \(e/(e-1)\)-approximation algorithm [24]. It is interesting to see if our weight transfer techniques (or its variants) can be applied here to give a PTAS for this problem.

A particularly interesting milestone is the following special case: Suppose that there is a feasible solution of cost \(k\) that protects all terminals in \(T\). Can we find a solution of cost \(k\) that protects, e.g., \((1-\epsilon)\) fraction of terminals in \(T\)? This case is, in some sense, a special case of the Firefighters problem with “perfect completeness” condition.

5. Motivated by RMFC on trees, we define a new problem that we call **Minimum Congestion Set Cover problem**, defined as follows. There is a ground set \([n]\) and a set system \(\mathcal{S} = \{S_1, \ldots, S_m\}\) where \(S_i \subseteq [n]\). The sets are partitioned into several groups by a set \(\mathcal{G}\). Our objective is to find a sub-collection of sets \(\mathcal{S}' \subseteq \mathcal{S}\) such that, for each group \(X \in \mathcal{G}\), \(|X \cap \mathcal{S}'| \leq k\) and \(\mathcal{S}'\) covers all elements in the ground set, while minimizing \(k\).

There is a connection between this problem to many covering and related problems, mostly in the context where the set systems have certain structures. RMFC on trees can be viewed as a special case of this problem. It is an interesting research direction to characterize the properties that allow us to do weight transfer scheme and extend it to this general set systems setting.
REFERENCES


[59] C. Hendler. Schranken fur farbungs-und cliquemuberdeckungszahl geometrisch represen


[62] David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-


197


