ROUTING ALGORITHMS FOR RING NETWORKS

by

Yong Wang

B.Sc., Peking University, 1999

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
July 2003

APPROVAL

Name: Yong Wang

Degree: Master of Science

Title of thesis: Routing Algorithms for Ring Networks

Examining Committee: Dr. Petra Berenbrink
Chair

Dr. Qian-ping Gu
Senior Supervisor

Dr. Joseph G. Peters
Supervisor

Dr. Arthur L. Liestman
SFU Examiner

Date Approved: July 31, 2003

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENSE

Title of Thesis/Project/Extended Essay:

Routing Algorithms for Ring Networks

Author:

(signature)

Yong Wang

(name)

Aug. 6, 2003

(date)

Abstract

In this thesis, we study routing problems on ring networks. The ring is a popular topology for communication networks and has attracted much research attention.

A communication application on a ring network can be regarded as a set of connection requests, each of which is represented by a set of nodes to be connected in the ring network. To implement a communication application, we need to develop a routing algorithm to find a path connecting all the nodes involved in each connection request. One of the most important optimization problems for the communication on ring networks is to develop a routing algorithm such that the maximum congestion (i.e., the maximum number of paths that use any single link in the ring) is minimized. This problem can be formulated as the *Minimum Congestion Hypergraph Embedding in a Cycle (MCHEC)* problem with a set of connection requests represented by a hypergraph. A special case of the MCHEC problem, in which each connection request involves exactly two nodes, is known as the *Minimum Congestion Graph Embedding in a Cycle* problem. A more general case, in which connection requests may have non-uniform bandwidth requirements, is known as the *Minimum Congestion Weighted Hypergraph Embedding in a Cycle* problem. The Minimum Congestion Graph Embedding in a Cycle problem is solvable in polynomial time, and the other problems are NP-hard.

In this thesis, we focus on the MCHEC problem and propose efficient algorithms in three categories. In the first category is a 1.8-approximation algorithm that improves the previous 2-approximation algorithms. In the second category is an algorithm that computes optimal solutions for the MCHEC problem. This algorithm runs in polynomial time for subproblems with constant maximum congestions, and is more efficient in terms of the time complexity than the previous algorithm that solves the same problem. The third category contains two heuristic approaches. According to our simulation results, both heuristics have lower time complexities and better practical performance than a well known heuristic.

To my wife, my parents and my little sister

淡泊明志

宁静致远

------ 诸葛亮 《诫子篇》

Acknowledgments

I would like to express my deepest thanks to my senior supervisor, Dr. Qian-ping Gu, for his valuable guidance, continuous encouragement, and instructions on academic writing throughout this research. I am very thankful to my supervisor, Dr. Joseph G. Peters, who is the organizer of Network Modeling Group. I learned much through our group meetings. I would also like to thank Dr. Arthur L. Liestman for serving as the examiner of this thesis and being very responsible. He gave me many valuable advices on academic writing.

I would like to take this opportunity to say thanks to all the friends whom I have met at Simon Fraser University, and all the staff and faculty in the School of Computing Science, for making a nice study and working environment. Special thanks go to Dr. Pavol Hell, I gained a lot in his class.

Last but not least, I am deeply in debt to my parents for their continuous love. I would like to express my great gratitude to my wife for her love and encouragement all the time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We study routing problems on ring networks in this thesis. In a ring network, a set of network nodes are connected together by a set of links as a cycle, and every node plays the same role. This node-symmetry simplifies the design of network algorithms such as routing and path coloring. Moreover, a ring is a 2-connected topology. There exist two distinct paths between any pair of nodes in a ring network, so it remains connected even in the presence of any single node or link failure. Taking account of these advantages over other network configurations, the ring topology is widely used in communication networks and has attracted much attention as a research subject [1, 2, 3, 4, 5, 6, 7, 8].

A communication application on a ring network can be regarded as a set of connection requests, each of which is represented by a set of nodes to be connected in the ring network. To implement a communication application, we need to develop a routing algorithm to find a path connecting all the nodes involved in each connection request. Researchers studied routing problems on ring networks either as directed connection requests on directed ring networks [5] or as undirected connection requests on undirected ring networks [4]. In the former case, a ring network is modeled as a *symmetric digraph* with two directed links in two opposite directions between every pair of adjacent nodes. In the latter case, a ring network is modeled as an *undirected graph* with one undirected link between every pair of adjacent nodes. In some practical ring networks such as optical rings, current technologies do not support the bi-directional use of a link, so two opposite-directed optical fibers are usually used to connect each pair of adjacent nodes. Therefore, the directed model is more appropriate to describe such ring networks. However, the undirected model is commonly used in studying communication problems because it is simple and appropriate for modeling

two-way communications. Results derived from the undirected model can often be easily extended to practical ring networks with directed communication links. In this thesis, the discussion is based on the undirected model. As a part of the future work, we will extend the results of this thesis to the directed model.

We assume that all the links in a ring network have the same capacity, which is also referred to as the *capacity* of the ring. The cost of a ring network depends on its capacity. Therefore, an important optimization problem arises in designing routing algorithms for ring networks: Given a set of connection requests, develop a routing algorithm to find a path in the ring for each connection request such that the required ring capacity to satisfy all the connection requests is minimized. If we define the *congestion* of a link to be the number of paths containing the link and the *maximum congestion* of a ring to be the maximum number of paths containing any single link of the ring, then the required ring capacity is equal to the maximum congestion on the ring network.

A communication application consisting of m connection requests on a ring network with n nodes can be described by a hypergraph with m hyperedges and n nodes. Each hyperedge in the hypergraph represents a connection request among its nodes. Figure 1.1(a) shows a hypergraph that corresponds to four connection requests $\{0,1,2\}$, $\{0,4,5\}$, $\{3,4,5\}$, and $\{3,5\}$ on node set $\{0,1,2,3,4,5\}$. Since routing a set of connection requests on a ring network can be regarded as embedding corresponding hyperedges as paths in a cycle, Ganley and Cohoon [9] formulated the above routing problem on ring networks as the *Minimum Congestion Hypergraph Embedding in a Cycle (MCHEC)* problem: Embed hyperedges of a hypergraph as paths in a cycle such that the maximum congestion is minimized. Figure 1.1(b) and (c) give embeddings with maximum congestion 3 and maximum congestion 2 respectively for the hypergraph in Figure 1.1(a). A special case of the MCHEC problem, in which each connection request involves exactly two nodes, is in fact the *Minimum Congestion Graph Embedding in a Cycle* problem. The Minimum Congestion Graph Embedding in a Cycle problem has been proved to be solvable in polynomial time by Frank et al. [3]. However, the MCHEC problem is known to be NP-complete [9]. Ganley and Cohoon [9] gave a 3-approximation algorithm for the MCHEC problem. The approximation ratio was improved to 2 based on different approaches [10, 11, 12]. Gonzalez [10] proposed two 2-approximation algorithms for the MCHEC problem. One algorithm formulated the MCHEC problem as an *Integer Program*, and the other transformed the MCHEC problem to the Minimum Congestion Graph Embedding in a Cycle problem. Carpenter et al. [11] gave

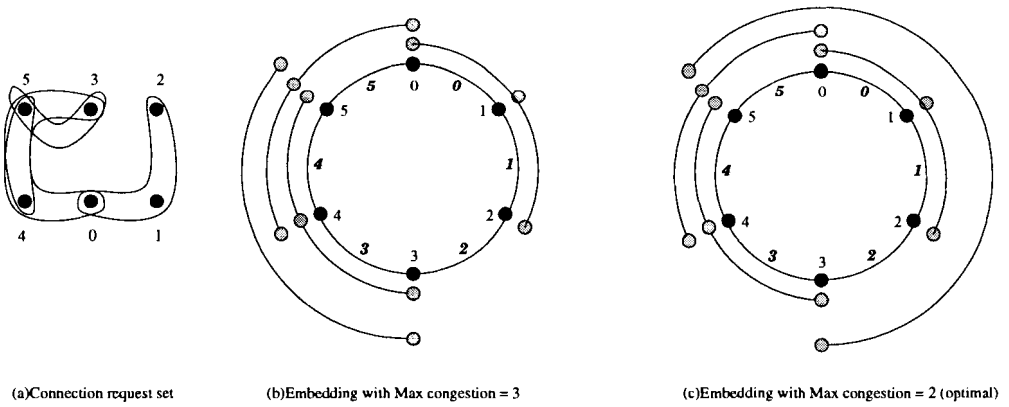

Figure 1.1: Hypergraph embedding in a cycle

a simple 2-approximation algorithm called *Clockwise Embedding*. Lee and Ho [12] developed a greedy algorithm named *Longest Adjacent Path Removing* that has an approximation ratio of 2 as well. Ganley and Cohoon [9] also proposed an $O((mn)^{L^*+1})$ time algorithm to compute optimal solutions for the MCHEC problem, where L^* is the maximum congestion, m is the number of hyperedges, and n is the number of nodes in the hypergraph. This algorithm runs in polynomial time for the subproblem with constant maximum congestion L^*. In addition, some heuristic algorithms have been proposed to achieve good performance in practice. Among them, the *Iterated Maximum Independent Set (IMIS)* heuristic presented by Ganley and Cohoon [13] is very well known.

For the MCHEC problem discussed above, we assume that the connection requests have a uniform bandwidth requirement, so the congestion of each link in a cycle can be simplified to be the number of paths using the link. A more general case, in which the connection requests may have non-uniform bandwidth requirements, was formulated by Lee and Ho [12] as the *Minimum Congestion Weighted Hypergraph Embedding in a Cycle* problem: Embed the weighted hyperedges of a hypergraph as weighted paths in a cycle such that the maximum congestion (i.e., the maximum total weight of paths using any single link in the cycle) is minimized. Lee and Ho proved the Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem is NP-complete and gave two 2-approximation algorithms [12]. Again for a special case in which each connection request involves exactly two nodes, the Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem is in fact the *Minimum Congestion Weighted Graph Embedding in a Cycle* problem. The Minimum Congestion

Weighted Graph Embedding in a Cycle problem is proved to be NP-complete as well by Cosares and Iraj [14]. The approximation algorithms has been proposed in [14, 7, 15].

In this thesis, we concentrate on the MCHEC problem. The MCHEC problem has applications not only in network communications, but also in parallel computing and electronic design automation [3, 13, 16, 17, 18, 19]. In parallel computing, the processors of a parallel computer can be represented by the nodes of a hypergraph; a group of processors, which are required to communicate with one another, can be represented by a hyperedge in the hypergraph. A solution to the MCHEC problem gives a routing path for each communication group. In electronic design, among applications is the moat routing. An instance of the moat routing consists of two concentric rectangles, which represent a core circuit area and input/output pads respectively, and a set of nets whose pins lie on either or both of the perimeters of the two rectangles. The routing of each net will be placed between the two rectangles. The goal is to implement the routing for the set of nets such that the width of the circular channel between the two rectangles is minimized.

We propose algorithms for the MCHEC problem in three categories: the approximation algorithm, the algorithm for optimal solutions, and the heuristic approach. In the first category is a 1.8-approximation algorithm that improves the best known 2-approximation algorithms. Our algorithm starts from the Clockwise Embedding, and then the algorithm tries to re-embed some hyperedges to reduce the maximum congestion. Let L be the maximum congestion of the Clockwise Embedding and L^* be the maximum congestion of an optimal embedding. If our algorithm can re-embed k hyperedges to get an embedding with maximum congestion $L-k$, the approximation ratio of the algorithm is $(L-k)/L^*$. Since $\lceil L/2 \rceil \leq L^*$ [11], the approximation ratio of our algorithm is at most $2(L-k)/L$. This gives a good approximation ratio if k is large. If k is small, we shall prove a new lower bound on L^*. As shown later, the approximation ratio of the algorithm increases from 1 to 1.8 as k decreases from $L/2$ to $L/10$, the ratio decreases from 1.8 to 1.5 as k decreases from $L/10$ to 0, and the ratio is 1.8 in the worst case. Our algorithm has the optimal $O(mn)$ time for the hypergraph with m hyperedges and n nodes.

In the second category is an $O((\sqrt{m}n)^{L^*})$ time algorithm to compute optimal solutions for the MCHEC problem, where L^* is the maximum congestion, m is the number of hyperedges, and n is the number of nodes in the hypergraph. This algorithm improves the $O((mn)^{L^*+1})$ time algorithm given by Ganley and Cohoon [9], and runs in polynomial time

for the subproblem with constant maximum congestion L^*. The $O((mn)^{L^*+1})$ time algorithm uses the fact that every link in a cycle has a congestion at most L^* in any optimal embedding. An arbitrary link i in the cycle is chosen, and an optimal embedding is sought by enumerating all the "good" embeddings that incur a congestion of at most L^* on link i. An important observation is that the fewer "good" embeddings, the more efficient the algorithm. So in our algorithm, instead of considering only one link i, we think of two links in the cycle and merely check those "good" embeddings that incur congestions of at most L^* on both of the two links. Therefore, our algorithm reduces the number of checked "good" embeddings further, and thus improves the efficiency of the algorithm.

The third category contains two heuristics. One heuristic uses a greedy strategy, and the other tries to evenly distribute congestions on all the links in a cycle. Both heuristics run in $O(mn)$ time, which improves the $O(m(mn)^2)$ time Iterated Maximum Independent Set heuristic proposed by Ganley and Cohoon [13] for the hypergraph with m hyperedges and n nodes. In addition, our heuristics achieve much better performance than the Iterated Maximum Independent Set heuristic in practice according to our simulation results.

This thesis is organized as follows. In chapter 2, we introduce basic concepts and notation. In chapter 3, the previous work about the MCHEC problem is surveyed. Chapter 4 gives our improved algorithms for the MCHEC problem in three categories. In the last chapter, we present possible research directions for the future work.

Chapter 2

Preliminaries

2.1 Approximation algorithm and approximation ratio

For any NP-hard optimization problem, there is no polynomial time algorithm to compute an optimal solution unless $P = NP$. So much effort has been put into developing polynomial time algorithms to find near-optimal solutions whose values are close to the value of an optimal solution. We call such a near-optimal solution an *approximate solution*, and an algorithm that produces approximate solutions an *approximation algorithm.* The following formal definitions of the approximate solution, approximation algorithm, and approximation ratio are from the book by Cormen et al. [20].

Definition 2.1.1 *[20] (Approximate Solution) For an optimization problem* $\prod$*, a feasible solution with the value close to the value of an optimal solution is an approximate solution of* $\prod$.

Definition 2.1.2 *[20] (Approximation Algorithm) For an optimization problem* $\prod$*, a polynomial time algorithm that generates approximate solutions is an approximation algorithm of* $\prod$.

Any algorithm that produces approximate solutions for problem $\prod$ can be called an approximation algorithm. How to evaluate approximation algorithms is important. Generally speaking, good approximation algorithms should be efficient (polynomial time) and produce solutions with values as close to the optimum as possible. Concept *approximation ratio* [20] is widely used to evaluate approximation algorithms. In what follows, we assume that a solution always has a positive value.

Definition 2.1.3 *[20] (Approximation Ratio) An approximation algorithm A for an optimization problem $\prod$ has an approximation ratio of $\rho(n)$ if for any instance of $\prod$ with size n, the value C of any approximate solution produced by the approximation algorithm satisfies*

$$\max\{\frac{C}{C^*}, \frac{C^*}{C}\} \leq \rho(n),$$

where C^* is the value of an optimal solution.

This definition applies for both maximization and minimization problems. For a maximization problem, $0 < C \leq C^*$, and C^*/C gives the approximation ratio. For a minimization problem, $0 < C^* \leq C$, and C/C^* gives the approximation ratio. Therefore, the approximation ratio of an approximation algorithm is at least 1, and the approximation ratio of an optimal algorithm is exactly 1. An approximation algorithm with a large approximation ratio may return a solution that is much worse than an optimal solution. Reducing the approximation ratio as close to 1 as possible is a major goal of developing approximation algorithms.

2.2 Notation and terminology

A *cycle* C of n nodes is an undirected graph with node set $\{i \mid 0 \leq i \leq n-1\}$. There is a link between nodes i and j if $i = j \pm 1$, where (and in what follows) the arithmetic involving nodes is performed implicitly using the modulo n operation. The link from node i to node $i+1$ is labeled as link i (see Figure 1.1). In what follows, without loss of generality, we assume that n nodes and n links of C are labeled from 0 to $n-1$ in the clockwise direction as shown in Figure 1.1(b) and (c).

A *hypergraph* $H(V, E_H)$ of n nodes and m hyperedges is a hypergraph with node set $V = \{i \mid 0 \leq i \leq n-1\}$ and hyperedge set $E_H = \{e_1, e_2, \ldots, e_m\}$, where each hyperedge e_i is a subset of V with n_i ($n_i \geq 2$) nodes. As a special case, a *graph* $G(V, E)$ is a hypergraph with each hyperedge consisting of exactly two nodes (i.e., $n_i = 2$ for every e_i in graph $G(V, E)$). For a hypergraph $H(V, E_H)$, let $\{v_1^i, v_2^i, \ldots, v_{n_i}^i\}$ be the sorted set of nodes contained in hyperedge e_i (i.e., $v_1^i < v_2^i < \ldots < v_{n_i}^i$). For $1 \leq i \leq m$, a *connecting path* (or *c-path*) P_i in C for hyperedge e_i is a path in C such that all the nodes in e_i are connected by P_i. An *adjacent path* (or *a-path*) in C for hyperedge e_i is a path in C from node v_j^i to node v_{j+1}^i ($1 \leq j \leq n_i$) in the clockwise direction, where $v_{n_i+1}^i = v_1^i$. An embedding of hypergraph

$H(V, E_H)$ in cycle C is a set of c-paths in C such that there is exactly one c-path in the set for each hyperedge. A hyperedge e_i with n_i nodes has exactly n_i a-paths in C, and any $n_i - 1$ a-paths together form a path in C to connect all the n_i nodes in e_i. Thus any $n_i - 1$ a-paths constitute a c-path P_i for hyperedge e_i. Given an embedding of a hypergraph, the congestion of each link in C is the number of c-paths that contain the link. More precisely, we define the MCHEC problem as follows: Given a hypergraph and a cycle on the same node set, embed the hyperedges as c-paths in the cycle such that the maximum congestion (i.e., the maximum number of c-paths using any single link in the cycle) over all the links in the cycle is minimized. Figure 1.1(b, c) gives two feasible embeddings for the hypergraph in Figure 1.1(a) on the cycle with node set $\{0, 1, 2, 3, 4, 5\}$. The optimal embedding may not be unique, and Figure 2.1(c) is one of such embeddings.

A *segment* of cycle C is a connected subgraph of C. In what follows, we use *segment* $\langle p, q \rangle$ to denote the segment of C that includes nodes $p, p+1, \ldots, q$. Cycle C is cut into two segments by removing any two links i and j, and we call the two links i and j a *cut* (i, j) of C. A hyperedge is *separated* by a cut if there is at least one node of the hyperedge in each of the two segments. For example, if we choose links 0 and 3 as a cut of the cycle in Figure 1.1, hyperedges $e_1 = \{0, 1, 2\}, e_3 = \{3, 4, 5\}$, and $e_4 = \{3, 5\}$ are separated by cut $(0, 3)$; hyperedge $e_2 = \{0, 4, 5\}$ is not separated by cut $(0, 3)$. It is observed that the c-path for a separated hyperedge in any embedding must contain at least one of the two links in the cut. Therefore, if we use $N(i, j)$ to denote the number of hyperedges separated by cut (i, j), then $\max_{i,j}\{\frac{N(i,j)}{2}\}$ is a lower bound on the maximum congestion of an optimal embedding.

Chapter 3

Related work on the MCHEC problem

3.1 A special case: Minimum Congestion Graph Embedding in a Cycle

Frank et al. [3] proved the following theorem for the Minimum Congestion Graph Embedding in a Cycle problem:

Theorem 3.1.1 *[3] The Minimum Congestion Graph Embedding in a Cycle problem can be solved optimally in polynomial time.*

Schrijver et al. [7] developed an $O(mn^2)$ time optimal algorithm for the graph with m edges and n nodes.

3.2 Minimum Congestion Hypergraph Embedding in a Cycle (MCHEC)

3.2.1 NP-completeness

As a general case of the Minimum Congestion Graph Embedding in a Cycle problem, the MCHEC problem deals with the situation in which each connection request may involve more than two nodes. The MCHEC problem is more complicated than the Minimum Congestion Graph Embedding in a Cycle problem, and has been proved to be NP-complete by Ganley

and Cohoon [9]. They first transformed the MCHEC problem to the following *Cycle Cover by Multiple Choice Paths (CCMCP)* problem: Given the same problem instance as the MCHEC problem, find a path in the cycle for each hyperedge to connect two nodes in the hyperedge without spanning any other node in the same hyperedge, such that the minimum congestion over all the links in the cycle is maximized. For any solution of the CCMCP problem, we can easily transform it to a solution of the MCHEC problem by setting each path in the solution of the CCMCP problem to be the complement of the path. Therefore, if the CCMCP problem is NP-complete, so is the MCHEC problem. To prove the NP-completeness of the CCMCP problem, Ganley and Cohoon constructed a polynomial time transformation from a known NP-complete problem *Numerical Matching with Target Sums (NMTS)* [21] to the CCMCP problem. We refer to Ganley and Cohoon [9] for a detailed proof of the following theorem.

Theorem 3.2.1 *[9] The MCHEC problem is NP-complete.*

3.2.2 Approximation algorithms

1. *A simple 3-approximation algorithm*

   Recall that any two links in a cycle C form a cut of C. Ganley and Cohoon [9] presented a simple algorithm for the MCHEC problem as follows: Choose an arbitrary cut (i, j) consisting of links i and j. Let S denote the set of hyperedges that are separated by cut (i, j), and $\bar{S}$ denote the set of hyperedges that are not separated by cut (i, j). Embed every hyperedge in $\bar{S}$ within the segment containing its nodes, and embed the hyperedges in S arbitrarily. This algorithm has an approximation ratio of 3 and runs in $O(mn)$ time for the hypergraph with m hyperedges and n nodes.

2. *A 2-approximation algorithm based on Linear Programming*

   Gonzalez formulated the MCHEC problem as an *Integer Program* [10]. As is well known, solving an Integer Program is NP-hard, however, a *Linear Program* is solvable in polynomial time. Gonzales transformed the Integer Program to a corresponding Linear Program by performing the *LP Relaxation*, and then obtained an approximate solution of the Integer Program by rounding off the solution of the Linear Program. This algorithm has an approximation ratio of 2. However, the running time of the algorithm is dominated by the part computing Linear Program, which has a high time complexity.

3. *A 2-approximation algorithm based on the Minimum Congestion Graph Embedding in a Cycle problem*

   As an alternative to the above *Linear Programming (LP)* based approximation algorithm, Gonzales proposed a *LP-Free* algorithm with a lower time complexity [10].

   Recall that the Minimum Congestion Graph Embedding in a Cycle problem can be solved optimally in polynomial time [3]. The LP-Free algorithm transformed any instance of the MCHEC problem to a corresponding instance of the Minimum Congestion Graph Embedding in a Cycle problem. Once an optimal embedding of the Minimum Congestion Graph Embedding in a Cycle problem is obtained, it is then transformed back to an approximate solution of the original MCHEC problem.

   The LP-Free algorithm has an approximation ratio of 2 as well. For the hypergraph with m hyperedges and n nodes, the time complexity is $O(mn^3)$, which is dominated by the time complexity of the optimal algorithm for the Minimum Congestion Graph Embedding in a Cycle problem.

4. *A 2-approximation algorithm based on the Clockwise Embedding*

   Carpenter et al. proposed a very simple approximation algorithm named *Clockwise Embedding* [11]. The idea is to embed each hyperedge as a path from its lowest numbered node to its highest numbered node in the clockwise direction. Figure 3.1 shows an example of the Clockwise Embedding. For hyperedge $e_1 = \{0, 1, 2\}$, the lowest numbered node is 0 and the highest numbered node is 2. So e_1 is embedded as the path from node 0 to node 2 in C in the clockwise direction. The other three hyperedges are embedded similarly. In the Clockwise Embedding, no c-paths contain the highest numbered link, which is link 5 in Figure 3.1.

   The Clockwise Embedding is also a 2-approximation algorithm. Since our improved approximation algorithm will be based on the Clockwise Embedding, we state the complete proof by Carpenter et al. [11] for the following theorem.

   Theorem 3.2.2 *[11] The approximation ratio of the Clockwise Embedding is 2.*

   Proof: Recall that $N(i, j)$ denotes the number of hyperedges separated by cut (i, j) of a cycle C, and $\max_{i,j}\{\frac{N(i,j)}{2}\}$ is a lower bound on the maximum congestion of an optimal embedding. Let $l(i)$ be the congestion of any link i in C in the Clockwise Embedding.

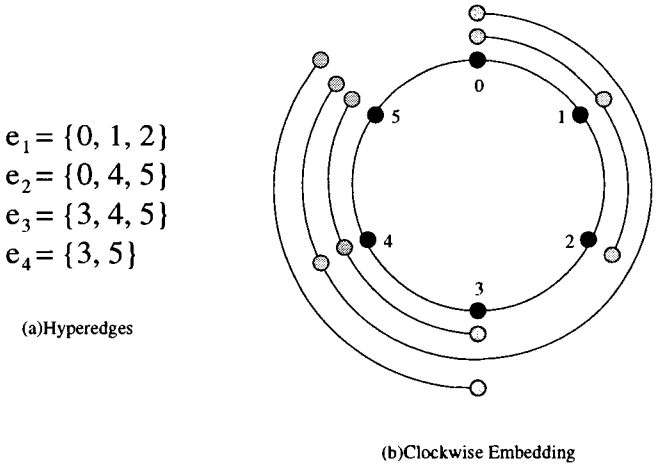

Figure 3.1: Clockwise Embedding

Let $L = \max_i\{l(i)\}$, and s be a link with maximum congestion L in the Clockwise Embedding. Since no c-paths contain link $n-1$ (i.e., the highest numbered link in C), all the hyperedges whose c-paths contain link s in the Clockwise Embedding are separated by cut $(s, n-1)$. Therefore, $L = N(s, n-1) \leq 2 * \max_{i,j}\{\frac{N(i,j)}{2}\} \leq 2L^*$, that is, the maximum congestion of the Clockwise Embedding is bounded above by twice that of an optimal embedding. □

The Clockwise Embedding runs in $O(mn)$ time for the hypergraph with m hyperedges and n nodes.

5. *A 2-approximation algorithm based on the Longest Adjacent Path Removing*

   Lee and Ho proposed a simple approximation algorithm called *Longest Adjacent Path Removing* [12]. Every hyperedge e_i with n_i nodes has n_i a-paths in cycle C. Any $n_i - 1$ a-paths together constitute a c-path of hyperedge e_i. Therefore, if any one of the n_i a-paths is removed, the rest $n_i - 1$ a-paths form a feasible embedding for hyperedge e_i. The *Longest Adjacent Path Removing* algorithm simply deletes the longest a-path for each hyperedge.

   The Longest Adjacent Path Removing has an approximation ratio of 2. It runs in $O(mn)$ time for the hypergraph with m hyperedges and n nodes.

3.2.3 An algorithm for optimal solutions

For any fixed integer k and hypergraph with m hyperedges and n nodes, Ganley and Cohoon proposed an algorithm to compute solutions for the MCHEC problem with maximum congestion at most k, or to determine that such solutions do not exist in $O((mn)^{k+1})$ time.

Their algorithm was stated as follows: Choose an arbitrary link i in cycle C. For every subset S of hyperedges with $|S| \leq k$, check all the possible embeddings such that link i is used by all the hyperedges in S, and link i is not used by any hyperedge that is not in S. In the worst case, each hyperedge in S has $n-1$ possible embeddings that use link i, and each hyperedge that is not in S has only one unique embedding that does not use link i. Therefore in the worst case, the total number of checked embeddings is

$$\sum_{i=1}^{k} \binom{m}{i} (n-1)^i = O((mn)^k).$$

Embedding the hyperedges requires another $O(mn)$ time, so the algorithm runs in $O((mn)^{k+1})$ time. This algorithm can be used to construct an optimal algorithm running in $O((mn)^{L^*+1})$ time for the MCHEC problem, where L^* is the maximum congestion. For the subproblem with constant maximum congestion L^*, the optimal algorithm runs in polynomial time.

3.2.4 A heuristic

Ganley and Cohoon proposed a heuristic approach called *Iterated Maximum Independent Set (IMIS)* [13]. They claimed that the heuristic performed very well in empirical studies.

In a hypergraph with m hyperedges and n nodes, each hyperedge e_i consisting of n_i sorted nodes $\{v_1^i, v_2^i, \ldots, v_{n_i}^i\}$ has n_i possible c-paths (only those consisting of exactly $n_i - 1$ a-paths are considered). Each of these n_i c-paths can be considered as an arc in cycle C. Notice that if $n_i > 2$, all the arcs from the same hyperedge e_i are pairwise intersecting. For all the arcs from all the hyperedges, construct a *circular arc graph* G_{ca}, in which each node represents an arc and there is an edge between two nodes if the two corresponding arcs intersect in the cycle.

Compute a *maximum independent set (MIS)* for G_{ca} in $O((mn)^2)$ time using the algorithm proposed by Gupta et al. [22]. Since all the arcs from a hyperedge with $n_i > 2$ are pairwise intersecting, the MIS can contain at most one arc from each hyperedge. If the MIS contains both arcs from a hyperedge with $n_i = 2$, the size of the MIS must be 2, so an MIS that does not contain two arcs from the same hyperedge can be found exhaustively in

$O(m^2)$ time. Thus, computing an MIS that contains at most one arc from each hyperedge can be done in $O((mn)^2)$ time. Once an MIS is obtained, each hyperedge, for which there is an arc in the MIS, is embedded according to that arc. Remove all the arcs in G_{ca} that are from the embedded hyperedge, and repeat the same process until no arcs remain in G_{ca}.

An MIS is computed in $O((mn)^2)$ time for each round, and at most m rounds are required in the worst case. So the heuristic runs in $O(m(mn)^2)$ time.

Chapter 4

Efficient algorithms for the MCHEC problem

In this chapter, we develop new algorithms for the MCHEC problem in three categories. In the first category is a 1.8-approximation algorithm that improves the previous 2-approximation algorithms [10, 11, 12]. In the second category is an $O((\sqrt{mn})^{L^*})$ time algorithm to compute optimal solutions for the MCHEC problem, where L^* is the maximum congestion, m is the number of hyperedges, and n is the number of nodes in the hypergraph. This algorithm improves the previous $O((mn)^{L^*+1})$ time algorithm [9]. The third one contains two heuristics. Both of our heuristics not only have lower time complexities, but also beat the performance of the Iterated Maximum Independent Set heuristic [13] according to our simulation results.

4.1 A 1.8-approximation algorithm

4.1.1 Algorithm

In this section, we give a 1.8-approximation algorithm for the MCHEC problem. The basic idea of our algorithm is as follows: The algorithm starts from the Clockwise Embedding, and then re-embeds some hyperedges to reduce the maximum congestion of the Clockwise Embedding. Recall that L is the maximum congestion of the Clockwise Embedding and L^* is the maximum congestion of an optimal embedding. If our algorithm can re-embed k hyperedges to get an embedding with maximum congestion $L-k$, the approximation ratio of the algorithm is $(L-k)/L^*$. Since $\lceil L/2 \rceil \leq L^*$ (i.e., $\lceil L/2 \rceil$ is a lower bound on the maximum

congestion of an optimal embedding) [11], the approximation ratio of our algorithm is at most $2(L-k)/L$. If k is large, $2(L-k)/L$ gives a good approximation ratio. If k is small, we shall prove a new lower bound on L^* in order to achieve a good approximation as well. As shown later, the approximation ratio of our algorithm increases from 1 to 1.8 as k decreases from $L/2$ to $L/10$, and then decreases from 1.8 to 1.5 as k decreases from $L/10$ to 0. That is, the approximation ratio of our algorithm is always bounded above by 1.8. Furthermore, our algorithm has the optimal $O(mn)$ time for the hypergraph with m hyperedges and n nodes.

Re-embedding hyperedges decreases congestions on some links in cycle C, however, congestions on some other links could be increased as well. That is, re-embedding k hyperedges decreases the maximum congestion of the Clockwise Embedding by at most k. In our algorithm, we re-embed k hyperedges only when the re-embedding decreases the maximum congestion by exactly k. Thus our algorithm can re-embed at most $\lfloor L/2 \rfloor$ hyperedges (i.e., $0 \leq k \leq \lfloor L/2 \rfloor$), since $\lceil L/2 \rceil$ is a lower bound on the maximum congestion of an optimal embedding. To guarantee that re-embedding k hyperedges decreases the maximum congestion of the Clockwise Embedding by exactly k, we introduce the following terms.

Recall that $l(i)$ is the congestion of any link i in the Clockwise Embedding. For an integer k in the range $1 \leq k \leq \lfloor L/2 \rfloor$, let g_k be the lowest numbered link with $l(g_k) \geq L - 2k + 1$, and h_k be the highest numbered link with $l(h_k) \geq L - 2k + 1$. Thus $0 \leq g_k \leq h_k < n - 1$. For the example shown in Figure 4.1, $n = 10$, $L = 4$, $g_2 = 0$, $h_2 = 7$, $g_1 = 2$, and $h_1 = 6$. We call a hyperedge a *re-embedding candidate with respect to k (or candidate w.r.t. k)* if the hyperedge has a node in segment $\langle 0, g_k \rangle$, has a node in segment $\langle h_k + 1, n - 1 \rangle$, and has no node in segment $\langle g_k + 1, h_k \rangle$. Let x_k be the number of candidates w.r.t. k. In Figure 4.1, $x_2 = 0$, $x_1 = 1$, and hyperedge $\{0, 1, 8\}$ is a candidate w.r.t. $k = 1$. In our algorithm, we re-embed k hyperedges only when x_k, the number of candidates w.r.t. k, is greater than or equal to k. We re-embed any k out of the x_k candidates such that the c-path for each of the k candidates does not contain any link i in the range $g_k \leq i \leq h_k$ (see Figure 4.2). Therefore, re-embedding k candidates w.r.t. k decreases the maximum congestion by exactly k in our algorithm.

The definitions of g_k, h_k, and the candidate w.r.t. k imply that $g_{k+1} \leq g_k$, $h_{k+1} \geq h_k$, and any candidate w.r.t. $k+1$ is also a candidate w.r.t. k (i.e., $x_{k+1} \leq x_k$), where k is in the range $1 \leq k < \lfloor L/2 \rfloor$. To re-embed as many hyperedges as possible, our algorithm checks whether $x_k \geq k$ is true starting from $k = \lfloor L/2 \rfloor$. If it is true, our algorithm re-embeds

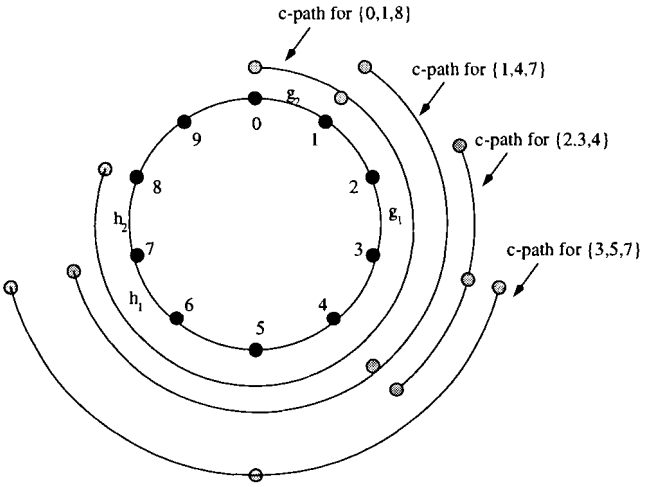

Figure 4.1: Re-embedding candidate

any k out of the x_k candidates to decrease the maximum congestion by k. Otherwise, our algorithm decreases k by one and repeats. Eventually, our algorithm terminates with k hyperedges re-embedded, where k is one of the values in $\{0, 1, \ldots, \lfloor L/2 \rfloor\}$.

The outline of our algorithm is as follows: We start with the Clockwise Embedding, and then we try to re-embed k candidates w.r.t. k to decrease the maximum congestion of the Clockwise Embedding by k. The re-embedding process starts from $k = \lfloor L/2 \rfloor$. If $x_k \geq k$ then we re-embed k or $k+1$ candidates and the algorithm terminates. Otherwise, k is decreased by one and the re-embedding process is repeated. As shown later, the algorithm has an approximation ratio of 1.8 except for a few special cases of fixed L. Although the $O((mn)^{L^*+1})$ time algorithm in [9] or the $O((\sqrt{m}n)^{L^*})$ time algorithm given in section 4.2 can be used to find optimal embeddings for the special cases, the time complexity is high in practice. We give a subroutine to handle those special cases. The subroutine is efficient and guarantees an approximation ratio of 1.8 for the special cases. Our algorithm and subroutine are given in Figure 4.3.

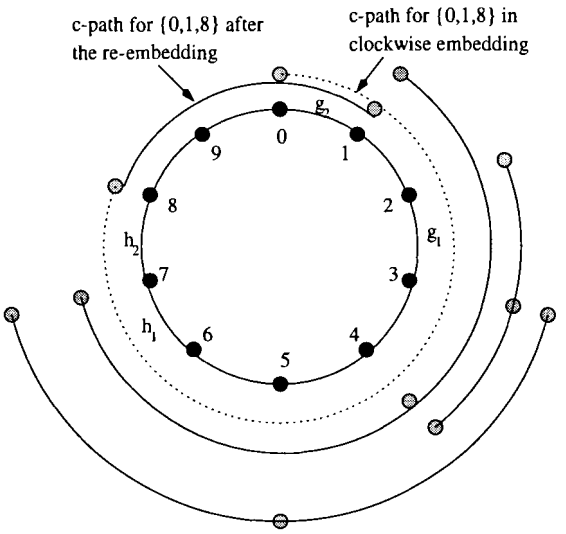

Figure 4.2: C-paths for the candidate w.r.t. $k = 1$ in the Clockwise Embedding and after the re-embedding.

4.1.2 Analysis

Algorithm R_Embedding terminates with k or $k+1$ hyperedges re-embedded, where k is one of the values in $\{0, 1, \ldots, \lfloor L/2 \rfloor\}$. Let L_k denote the maximum congestion achieved by our algorithm. The following lemma holds for L_k:

Lemma 4.1.1 *$L_k = L - k$ or $L_k = L - k - 1$.*

Proof: When algorithm R_Embedding terminates without calling Subroutine Special_Cases, either k candidates w.r.t. k are re-embedded, or $k+1$ candidates w.r.t. k, including at least one candidate w.r.t. $k+1$, are re-embedded. After a candidate w.r.t. k is re-embedded, the congestion of each link i with $g_k \leq i \leq h_k$ is decreased by one and the congestion of each link i with $i < g_k$ or $i > h_k$ is increased by at most one.

Assume that k candidates w.r.t. k are re-embedded. For each link i with $g_k \leq i \leq h_k$, the congestion of link i after the re-embedding is $l(i) - k \leq L - k$. For each link i with $i < g_k$ or $i > h_k$, the congestion of link i after the re-embedding is at most $l(i)+k \leq L-2k+k = L-k$. So after the re-embedding, maximum congestion L_k is $L - k$.

Procedure R_Embedding
Input: A hypergraph on the same node set of the cycle.
Output: An embedding of the hypergraph in the cycle.
begin

1. Perform the Clockwise Embedding for the hypergraph.
   Let L denote the maximum congestion of the Clockwise Embedding.
2. Find links g_k and h_k, and compute x_k for $k = 1, 2, ..., \lfloor L/2 \rfloor$.
   x_k is defined to be 0 for $k = \lfloor L/2 \rfloor + 1$ and $k = 0$.
3. $k := \lfloor L/2 \rfloor$.
   while $k \geq 1$ **do**
   if $(x_k \geq k)$ **then** goto step 4
   else $k := k - 1$.
4. **If** $x_k \geq k + 1$ **and** $x_{k+1} \geq 1$ **then**
   re-embed $k + 1$ arbitrary candidates w.r.t. k including
   at least one candidate w.r.t. $k + 1$
   else
   if $((L = 2$ **or** $L = 4)$ **and** $k = 0)$ **or** $(L = 12$ **and** $k = 1)$ **then**
   call Subroutine Special_Cases
   else re-embed k arbitrary candidates w.r.t. k.

end.

Subroutine Special_Cases
Input The Clockwise Embedding of the hypergraph.
Output An embedding of the hypergraph in the cycle.
begin
/* Let s be a link with maximum congestion L in the Clockwise Embedding,
E be the set of hyperedges whose c-paths contain link s in the Clockwise Embedding,
and P_i be the c-path for $e_i \in E$ that does not contain link s. */
If $(L = 2$ **or** $L = 4)$ **and** $k = 0$ **then**
for every $e_i \in E$ **do**
if re-embedding e_i as P_i reduces L by one **then**
re-embed e_i as P_i and **return**
else /* $L = 12$ **and** $k = 1$ */
for every pair $e_i, e_j \in E$ **do**
if re-embedding e_i as P_i and e_j as P_j reduces L by two **then**
re-embed e_i as P_i and e_j as P_j and **return**
re-embed an arbitrary candidate w.r.t. $k = 1$.
Return
end.

Figure 4.3: Embedding algorithm for the MCHEC problem.

Assume that $k+1$ candidates w.r.t. k are re-embedded. For each link i with $g_k \leq i \leq h_k$, the congestion of link i after the re-embedding is $l(i) - (k+1) \leq L - k - 1$. For each link i with $i < g_{k+1}$ or $i > h_{k+1}$, the congestion of link i after the re-embedding is at most $l(i) + (k+1) \leq L - 2(k+1) + (k+1) = L - k - 1$. For each link i with $g_{k+1} \leq i < g_k$ or $h_{k+1} \geq i > h_k$, since the $k+1$ re-embedded candidates include at least one candidate w.r.t. $k+1$, the congestion of link i after the re-embedding is at most $l(i) - 1 + k \leq L - 2k - 1 + k = L - k - 1$. So after the re-embedding, maximum congestion L_k is $L - k - 1$.

Assume that Subroutine Special_Cases is executed. For $k = 0$, $L_k = L - 1$ or $L_k = L$. For $k = 1$, $L_k = L - 2$ or $L - 1$. □

According to the above lemma, the approximation ratio of our algorithm is $(L-k)/L^*$ or $(L-k-1)/L^*$. If the algorithm terminates with a large k, we use $\lceil L/2 \rceil$ as a lower bound on L^* and get an approximation ratio of $(L-k)/\lceil L/2 \rceil$ or $(L-k-1)/\lceil L/2 \rceil$. This suggests that when terminating with a large k, our algorithm has a good approximation ratio. However, if the algorithm terminates with a small k, for example, $k = 0$, then the approximation ratio given by $L_k/\lceil L/2 \rceil$ is 2, which is no better than the Clockwise Embedding. In the following, we shall prove that if the algorithm terminates with a small k, then a new lower bound that is better than $\lceil L/2 \rceil$ can be found. Using this better lower bound, our algorithm has a good approximation ratio as well when k is small. Lower bound $\lceil L/2 \rceil$ is obtained by using a cut consisting of two links in the cycle. The new lower bound involves three links.

To derive the new lower bound, we need some new notation. Let x, y and z be any three distinct links in the cycle. Without loss of generality, we assume that $0 \leq x < y < z \leq n-1$. We define four disjoint subsets of hyperedges as follows:

W : the set of hyperedges such that each hyperedge has a node in segment $\langle z+1, x \rangle$, a node in segment $\langle x+1, y \rangle$, and a node in segment $\langle y+1, z \rangle$;

X : the set of hyperedges such that each hyperedge has a node in segment $\langle z+1, x \rangle$, has NO node in segment $\langle x+1, y \rangle$, and has a node in segment $\langle y+1, z \rangle$;

Y : the set of hyperedges such that each hyperedge has a node in segment $\langle z+1, x \rangle$, a node in segment $\langle x+1, y \rangle$, and has NO node in segment $\langle y+1, z \rangle$;

Z : the set of hyperedges such that each hyperedge has NO node in segment $\langle z+1, x \rangle$, has a node in segment $\langle x+1, y \rangle$, and a node in segment $\langle y+1, z \rangle$.

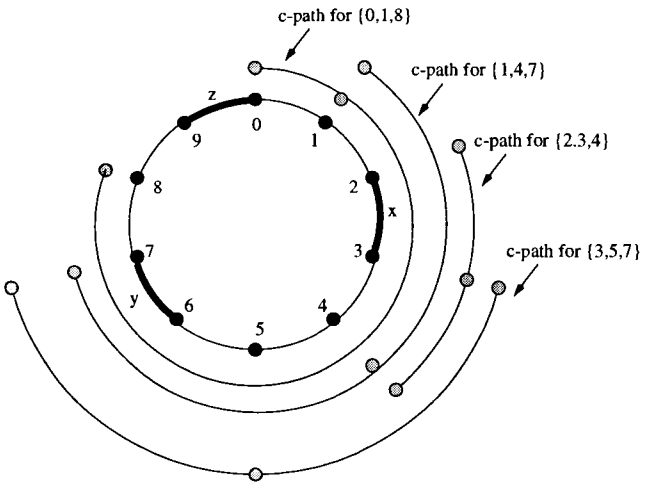

Figure 4.4: Lower bound involving three links x, y, and z

For the example shown in Figure 4.4, if we assume that $x = 2$, $y = 6$, and $z = 9$, then hyperedge $\{1, 4, 7\}$ belongs to set W, hyperedge $\{0, 1, 8\}$ belongs to set X, hyperedge $\{2, 3, 4\}$ belongs to set Y, and hyperedge $\{3, 5, 7\}$ belongs to set Z.

The intuition for proving the new lower bound involving three links is as follows: When algorithm R_Embedding terminates with some small k, for the two corresponding links g_k, h_k and the highest numbered link $n-1$, we get four disjoint subsets of hyperedges W, X, Y, and Z as defined above. The congestion of link g_k (resp. h_k) in the Clockwise Embedding is $l(g_k) \geq L - 2k + 1$ (resp. $l(h_k) \geq L - 2k + 1$). So when k is small, $l(g_k)$ (resp. $l(h_k)$) is large. A hyperedge whose c-path contains link g_k (resp. h_k) in the Clockwise Embedding belongs to one of the sets W, X, and Y (resp. W, X, and Z), that is, $l(g_k) = |W| + |X| + |Y|$ (resp. $l(h_k) = |W| + |X| + |Z|$). Therefore, when k is small, $|W| + |X| + |Y|$ (resp. $|W| + |X| + |Z|$) is large. Based on the above observations, a new lower bound can be obtained when k is small. The following lemma gives the new lower bound:

Lemma 4.1.2 *For any three links x, y and z with $0 \leq x < y < z \leq n-1$ in the cycle,*

$$L^* \geq \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

Proof: Let I denote an arbitrary embedding, and $l_I(i)$ denote the congestion on link i in embedding I. We have

$$L^* \geq \min_I\{\max\{l_I(x), l_I(y), l_I(z)\}\}.$$

Let $T = l_I(x) + l_I(y) + l_I(z)$. Notice that in any embedding I, the c-path for any hyperedge in W must contain at least two of links x, y, and z. Therefore, each hyperedge in W contributes at least 2 to T in any embedding I. Similarly, the c-path for any hyperedge in X, Y, or Z must contain at least one of links x, y, and z in any embedding I, so each hyperedge in X, Y, or Z contributes at least 1 to T.

Based on the above analysis, it is concluded that in any embedding I,

$$T \geq 2|W| + |X| + |Y| + |Z|.$$

So in any embedding I,

$$\max\{l_I(x), l_I(y), l_I(z)\} \geq \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

So

$$L^* \geq \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

□

It is worth pointing out that there exist hypergraphs for which the lower bound derived above is equal to the maximum congestion of an optimal embedding (i.e., the lower bound derived above is tight). An example of such hypergraphs is as follows: Let x, y, and z be three distinct links with $x < y < z$ in the cycle. We construct a hypergraph $H(V, E_h)$ with $E_h = W \cup X \cup Y \cup Z$. Each hyperedge in set X (resp. Y, Z) has nodes only in segment $\langle y+1, z+1\rangle$ (resp. $\langle z+1, x+1\rangle$, $\langle x+1, y+1\rangle$). W consists of hyperedges in three disjoint subsets W_x, W_y and W_z. Every hyperedge in W_x (resp. W_y, W_z) does not contain any node in segment $\langle z+2, x\rangle$ (resp. $\langle x+2, y\rangle$, $\langle y+2, z\rangle$). The sizes of sets W_x, W_y and W_z are defined as

$$\begin{aligned} |W_x| &= \frac{|W| - |X| + 2|Y| - |Z|}{3}, \\ |W_y| &= \frac{|W| - |X| - |Y| + 2|Z|}{3}, \quad \textit{and} \\ |W_z| &= \frac{|W| + 2|X| - |Y| - |Z|}{3}. \end{aligned}$$

Let I be an embedding for $H(V, E_h)$ such that the c-path for a hyperedge in W_j does not contain link j, where $j = x, y, z$; the c-path for a hyperedge in X (resp. Y, Z) does not contain x or y (resp. y or z, z or x). For any link i with $y + 1 \leq i \leq z$,

$$l_I(i) \leq |W| - |W_z| + |X| = \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

Similarly, for any link i with $z + 1 \leq i \leq x$,

$$l_I(i) \leq |W| - |W_x| + |Y| = \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|),$$

and for any link i with $x + 1 \leq i \leq y$,

$$l_I(i) \leq |W| - |W_y| + |Z| = \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

The proof for the 3-link lower bound can be easily extended to obtain lower bounds that involve more links. In this thesis, we only deal with the 3-link lower bound. Using the 3-link lower bound, we prove the following theorem:

Theorem 4.1.1 *The approximation ratio of algorithm R_Embedding is bounded above by 1.8.*

Proof:

According to Lemma 4.1.2,

$$L^* \geq \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|)$$

holds for three links g, h, and $n - 1$ with $0 \leq g < h < n - 1$.

Since $l(g) = |W| + |X| + |Y|$ and $l(h) = |W| + |X| + |Z|$, we have

$$\frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|) = \frac{1}{3}(l(g) + l(h) - |X|).$$

Therefore,

$$L^* \geq \frac{1}{3}(l(g) + l(h) - |X|). \tag{4.1}$$

Notice that for links g_k and h_k, if $g_k = h_k$ then g_k is the unique link with maximum congestion L. From the definition of x_k, we have $x_k = L$ when $g_k = h_k$.

Assume that algorithm R_Embedding terminates with maximum congestion L_k. The rest of the proof is divided into two cases:

Case 1: $L_k = L - k - 1$.

Taking $g = g_{k+1}$ and $h = h_{k+1}$ in inequality (4.1), we have $|X| = x_{k+1}$. Since the algorithm terminates with maximum congestion L_k, we have $x_{k+1} \leq k \leq \lfloor L/2 \rfloor$, which implies that $g_{k+1} < h_{k+1}$. Since $l(g_{k+1}) \geq L - 2(k+1) + 1$ and $l(h_{k+1}) \geq L - 2(k+1) + 1$,

$$\begin{aligned} L^* &\geq \frac{1}{3}\{l(g_{k+1}) + l(h_{k+1}) - x_{k+1}\} \\ &\geq \frac{1}{3}\{2(L - 2(k+1) + 1) - k\} \\ &= \frac{1}{3}(2L - 5k - 2). \end{aligned}$$

Therefore, an upper bound on the approximation ratio of the algorithm is

$$\frac{L_k}{L^*} \leq \frac{3(L-k-1)}{2L-5k-2}.$$

Since $\lceil L/2 \rceil$ is also a lower bound on L^*,

$$\frac{L_k}{L^*} \leq \frac{L-k-1}{\lceil L/2 \rceil}.$$

So the approximation ratio of the algorithm is bounded above by

$$\min\{\frac{L-k-1}{\lceil L/2 \rceil}, \frac{3(L-k-1)}{2L-5k-2}\}.$$

Function $(L-k-1)/\lceil L/2 \rceil$ is decreasing in k and function $3(L-k-1)/(2L-5k-2)$ is increasing in k. For $k \geq \lfloor L/10 \rfloor$, $(L-k-1)/\lceil L/2 \rceil \leq 1.8$; for $k \leq \lfloor L/10 \rfloor - 1$, $3(L-k-1)/(2L-5k-2) \leq 1.8$.

Case 2: $L_k = L - k$.

In this case, either $x_k = k$ or $x_{k+1} = 0$.

Assume that $x_k = k$. Taking $g = g_k$ and $h = h_k$ in inequality (4.1), then $|X| = x_k = k$ and $g_k < h_k$. Since $l(g_k) \geq L - 2k + 1$ and $l(h_k) \geq L - 2k + 1$,

$$\begin{aligned} L^* &\geq \frac{1}{3}\{l(g_k) + l(h_k) - x_k\} \\ &\geq \frac{1}{3}\{2(L - 2k + 1) - k\} \\ &= \frac{1}{3}(2L - 5k + 2). \end{aligned}$$

The approximation ratio of the algorithm is bounded above by

$$\min\{\frac{L-k}{\lceil L/2 \rceil}, \frac{3(L-k)}{2L-5k+2}\}.$$

For $k \geq \lceil L/10 \rceil$, $(L-k)/\lceil L/2 \rceil \leq 1.8$; for $k \leq \lceil L/10 \rceil - 1$, $3(L-k)/(2L-5k+2) \leq 1.8$.

Assume that $x_{k+1} = 0$. So

$$\begin{aligned} L^* &\geq \frac{1}{3}\{l(g_{k+1}) + l(h_{k+1}) - x_{k+1}\} \\ &\geq \frac{1}{3}\{2(L - 2(k+1) + 1)\} \\ &= \frac{1}{3}(2L - 4k - 2). \end{aligned}$$

The approximation ratio of the algorithm is bounded above by

$$\min\{\frac{L-k}{\lceil L/2 \rceil}, \frac{3(L-k)}{2L-4k-2}\}.$$

For $k \geq \lceil L/10 \rceil$, $(L-k)/\lceil L/2 \rceil \leq 1.8$; for $k \leq \lceil L/10 \rceil - 1$, $3(L-k)/(2L-4k-2) \leq 1.8$, except for the following three special cases:

$\langle 1 \rangle L = 2$ and $k = 0$,

$\langle 2 \rangle L = 4$ and $k = 0$, and

$\langle 3 \rangle L = 12$ and $k = 1$.

we shall prove that $L_k/L^* \leq 1.8$ holds for $\langle 1 \rangle$, $\langle 2 \rangle$, and $\langle 3 \rangle$ in the following.

Recall that s is a link with maximum congestion L in the Clockwise Embedding, E is the set of hyperedges whose c-paths contain link s in the Clockwise Embedding, and P_i is the c-path that does not contain link s for each $e_i \in E$. Let I_{opt} be an optimal embedding.

Case $\langle 1 \rangle$ can be divided into two sub-cases according to subroutine Special_Cases. Sub-case $\langle 1.1 \rangle$ is that one hyperedge $e_i \in E$ is re-embedded as P_i. So $L_k = L - 1 = 1$, which implies that $L_k/L^* = 1$. Sub-case $\langle 1.2 \rangle$ is that re-embedding any hyperedge in E does not decrease the maximum congestion. So no hyperedge is re-embedded and $L_k = L = 2$. We shall prove that 2 is a lower bound on L^* for sub-case $\langle 1.2 \rangle$, which implies that $L_k/L^* = 1$.

If any one of the following three conditions holds, then $L^* \geq 2$.

- *Condition 1:* There exists one hyperedge in E whose c-path in I_{opt} contains both links s and $n-1$.

  Notice that every hyperedge in E is separated by cut $(s, n-1)$. So the c-path for any $e_i \in E$ in any embedding must contain at least one of links s and $n-1$. If the condition 1 holds, then the sum of congestions on links s and $n-1$ in I_{opt} is at least 3, which implies that $L^* \geq 2$.

- *Condition 2:* There exist more than one hyperedges in E whose c-paths contain link s (or link $n-1$) in I_{opt}.

  This condition directly implies that $L^* \geq 2$.

- *Condition 3:* There exists one hyperedge that is not in E and whose c-path in I_{opt} is not the same as that in the Clockwise Embedding.

  For each hyperedge that is not in E, if it is not embedded in I_{opt} as in the Clockwise Embedding, its c-path must contain both links s and $n-1$ in I_{opt}. Therefore, the sum of congestions on links s and $n-1$ in I_{opt} is at least 4, which implies that $L^* \geq 2$.

Therefore, we can assume that none of the above three conditions holds. So the Clockwise Embedding can be transformed to I_{opt} by merely re-embedding one $e_i \in E$ as P_i. $L^* = 1$ means that re-embedding e_i as P_i decreases the maximum congestion of the Clockwise Embedding by one, which is a contradiction to the situation of sub-case $\langle 1.2 \rangle$. Thus $L^* \geq 2$, that is, $L_k/L^* = 1$.

Case $\langle 2 \rangle$ can also be divided into two sub-cases according to subroutine Special_Cases. Sub-case $\langle 2.1 \rangle$ is that one hyperedge $e_i \in E$ is re-embedded as P_i. So $L_k = L - 1 = 3$, which implies that $L_k/L^* \leq L_k/\lceil L/2 \rceil = 1.5 < 1.8$. Sub-case $\langle 2.2 \rangle$ is that re-embedding any hyperedge in E does not decrease the maximum congestion. So no hyperedge is re-embedded and $L_k = L = 4$. We shall prove that $L^* \geq 3$ for sub-case $\langle 2.2 \rangle$, which implies that $L_k/L^* \leq 1\frac{1}{3} < 1.8$.

If any one of the following three conditions holds, then $L^* \geq 3$.

- *Condition 1:* There exists one hyperedge in E whose c-path in I_{opt} contains both links s and $n-1$.

  Notice that every hyperedge in E is separated by cut $(s, n-1)$. So the c-path for any $e_i \in E$ in any embedding must contain at least one of links s and $n-1$. If the condition 1 holds, then the sum of congestions on links s and $n-1$ in I_{opt} is at least 5, which implies that $L^* \geq 3$.

- *Condition 2:* There exist more than two hyperedges in E whose c-paths contain link s (or link $n-1$) in I_{opt}.

  This condition directly implies that $L^* \geq 3$.

- *Condition 3:* There exists one hyperedge that is not in E and whose c-path in I_{opt} is not the same as that in the Clockwise Embedding.

  For each hyperedge that is not in E, if it is not embedded in I_{opt} as in the Clockwise Embedding, its c-path must contain both links s and $n-1$ in I_{opt}. Therefore, the sum of congestions on links s and $n-1$ in I_{opt} is at least 6, which implies that $L^* \geq 3$.

Therefore, we can assume that none of the above three conditions holds. So the Clockwise Embedding can be transformed to I_{opt} by merely re-embedding two $e_i's \in E$ as $P_i's$. $L^* = 2$ means that re-embedding any one of these two $e_i's$ as P_i decreases the maximum congestion of the Clockwise Embedding by one, which is a contradiction to the situation of sub-case $\langle 2.2 \rangle$. Thus $L^* \geq 3$, that is, $L_k/L^* \leq 1\frac{1}{3} < 1.8$.

There are two sub-cases for case $\langle 3 \rangle$ as well. Sub-case $\langle 3.1 \rangle$ is that two hyperedges $e_i, e_j \in E$ are re-embedded as P_i and P_j. So $L_k = L - 2 = 10$, which implies that $L_k/L^* \leq L_k/\lceil L/2 \rceil = 1\frac{2}{3} < 1.8$. Sub-case $\langle 3.2 \rangle$ is that re-embedding any two hyperedges in E does not decrease the maximum congestion by two. In this sub-case, only one candidate w.r.t. $k = 1$ is re-embedded in subroutine Special_Cases and $L_k = 11$. We shall prove that $L^* \geq 7$ for sub-case $\langle 3.2 \rangle$, which implies that $L_k/L^* \leq 1\frac{4}{7} < 1.8$.

If any one of the following three conditions holds, then $L^* \geq 7$.

- *Condition 1:* There exists one hyperedge in E whose c-path in I_{opt} contains both links s and $n-1$.

  Notice that every hyperedge in E is separated by cut $(s, n-1)$. So the c-path for any $e_i \in E$ in any embedding must contain at least one of links s and $n-1$. If the condition 1 holds, then the sum of congestions on links s and $n-1$ in I_{opt} is at least 13, which implies that $L^* \geq 7$.

- *Condition 2:* There exist more than six hyperedges in E whose c-paths contain link s (or link $n-1$) in I_{opt}.

  This condition directly implies that $L^* \geq 7$.

- *Condition 3:* There exists one hyperedge that is not in E and whose c-path in I_{opt} is not the same as that in the Clockwise Embedding.

For each hyperedge that is not in E, if it is not embedded in I_{opt} as in the Clockwise Embedding, its c-path must contain both links s and $n-1$ in I_{opt}. Therefore, the sum of congestions on links s and $n-1$ in I_{opt} is at least 14, which implies that $L^* \geq 7$.

Therefore, we can assume that none of the above three conditions holds. So the Clockwise Embedding can be transformed to I_{opt} by merely re-embedding six $e_i's \in E$ as $P_i's$. $L^* = 6$ means that re-embedding any $k(k \leq 6)$ of these six hyperedges $e_i's$ as $P_i's$ decreases the maximum congestion of the Clockwise Embedding by k. This is true especially for $k = 2$, which is a contradiction to the situation of sub-case $\langle 3.2 \rangle$. Thus $L^* \geq 7$, that is, $L_k/L^* \leq 1\frac{4}{7} < 1.8$. □

Step 1 of algorithm R_Embedding takes $O(mn)$ time for the hypergraph with m hyperedges and n nodes. Since $L = O(m)$, steps 2 and 4 can be done in $O(mn)$ time and step 3 can be done in $O(m)$ time. Subroutine Special_Cases takes $O(n)$ time. Therefore, the time complexity of algorithm R_Embedding is $O(mn)$. Notice that $O(mn)$ is also the optimal time to embed m hyperedges in the cycle with n nodes. The reason is as follows: For each hyperedge e_i with n_i nodes, it takes $\Omega(n_i)$ time to find any feasible embedding (i.e., construct a c-path) for e_i since a c-path for e_i consists of at least $n_i - 1$ a-paths. From the fact that n_i can be $\Omega(n)$ and there are m hyperedges in the hypergraph, we conclude that it takes $\Omega(mn)$ time to embed the hypergraph with m hyperedges and n nodes.

4.2 An algorithm for optimal solutions

Ganley and Cohoon [9] proposed an $O((mn)^{L^*+1})$ time algorithm to compute optimal solutions for the MCHEC problem, where L^* is the maximum congestion, m is the number of hyperedges, and n is the number of nodes in the hypergraph. Their algorithm runs in polynomial time for the subproblem with constant maximum congestion L^*. In this section, we present an $O((\sqrt{mn})^{L^*})$ time algorithm that solves the same problem.

First, for any fixed integer k and a hypergraph with m hyperedges and n nodes, we present an algorithm that computes a solution for the MCHEC problem with maximum congestion at most k, or determines that such a solution does not exist in $O((\sqrt{mn})^k)$ time. The intuition of this algorithm is as follows: To avoid checking all the possible $O(n^m)$ embeddings for the hypergraph with m hyperedges and n nodes, the algorithm in [9] arbitrarily selects a link in the cycle, and only checks the embeddings in which the congestion

on the selected link is at most k. Using this constraint, the algorithm need check $O((mn)^k)$ instead of $O(n^m)$ embeddings. Therefore, if we consider two links in the cycle, there will be two constraints for congestions on both links. Thus the number of checked embeddings can be reduced further.

Recall that L is the maximum congestion and s is a link with maximum congestion L in the Clockwise Embedding. Link $n-1$ is the highest number link in the cycle. Our algorithm considers these two links s and $n-1$. Let X be the set of hyperedges whose c-paths contain link s in the Clockwise Embedding, and Y be the set of hyperedges whose c-paths do not contain link s in the Clockwise Embedding. So $|X| = L$, and $|Y| = m - |X| = m - L$, where m is the total number of hyperedges. There are two key observations:

- *Observation 1:* Any c-path except the one in the Clockwise Embedding for each hyperedge in set X must contain link $n-1$, and may contain link s.
- *Observation 2:* Any c-path except the one in the Clockwise Embedding for each hyperedge in set Y must contain both links s and $n-1$.

Let I_{opt} be an optimal embedding and $l_{I_{opt}}(i)$ be the congestion on any link i in I_{opt}. In the Clockwise Embedding, assume that $t(t \leq m)$ hyperedges are not embedded as in I_{opt}, and assume that the other $m-t$ hyperedges are embedded as in I_{opt}. Among these t hyperedges that are not embedded as in I_{opt}, let $x(x \leq t)$ hyperedges are from set X, and the other $y = t - x$ hyperedges are from set Y.

Due to the above two observations,

$$l_{I_{opt}}(n-1) = x + y = t$$

and

$$l_{I_{opt}}(s) \geq (m-t) + y = ((L + |Y|) - (x + y)) + y = L + |Y| - x \geq L + y - x.$$

Since

$$l_{I_{opt}}(n-1) \leq k \quad and \quad l_{I_{opt}}(s) \leq k,$$

then

$$x + y \leq k \quad and \quad L + y - x \leq k.$$

Thus

$$y \leq k - \frac{L}{2} \leq \frac{k}{2}.$$

So if we start with the Clockwise Embedding, t hyperedges need to be re-embedded to obtain optimal embedding I_{opt}. Among these t hyperedges, x hyperedges are from set X, and y hyperedges are from set Y. Based on the above analysis, the number of possibilities to select t hyperedges to re-embed is:

$$\sum_{t=0}^{k} \sum_{y=0}^{\min\{t,\ k/2\}} \binom{m-L}{y} \binom{L}{t-y} = O((\sqrt{m})^k)$$

Each of these t selected hyperedges has $O(n)$ possible embeddings, so the total number of checked embeddings in our algorithm is $O((\sqrt{mn})^k)$.

Our $O((\sqrt{mn})^k)$ time algorithm can be stated formally as follows: For each set S of hyperedges with $|S| \leq k$, in which $y(y \leq \frac{k}{2})$ hyperedges belong to set Y and $x(x = |S| - y)$ hyperedges belong to set X, each hyperedge in S is embedded differently from the Clockwise Embedding. Each hyperedge that is not in S is embedded in the same way as the Clockwise Embedding. After checking all the possible embeddings, the algorithm either computes a solution with maximum congestion at most k, or determines that such a solution does not exist in $O((\sqrt{mn})^k)$ time.

Recall that L is the maximum congestion of the Clockwise Embedding, and the Clockwise Embedding is a 2-approximation algorithm. So maximum congestion L^* of an optimal embedding must be at least $\lceil L/2 \rceil$. Using the above $O((\sqrt{mn})^k)$ time algorithm, we construct an algorithm to compute optimal solutions for the MCHEC problem as follows: Starting from $k = \lceil L/2 \rceil$, the algorithm checks whether there exists a solution with maximum congestion at most k. If so, the algorithm terminates; otherwise, k is increased by one and the checking is repeated. The time complexity of the optimal algorithm is

$$O((\sqrt{mn})^{\lceil L/2 \rceil} + (\sqrt{mn})^{\lceil L/2 \rceil + 1} + (\sqrt{mn})^{\lceil L/2 \rceil + 2} + \cdots + (\sqrt{mn})^{L^*}) = O((\sqrt{mn})^{L^*}).$$

For the subproblem with constant maximum congestion L^*, the optimal algorithm runs in polynomial time.

4.3 Heuristics

Ganley and Cohoon [13] proposed the *Iterated Maximum Independent Set (IMIS)* heuristic, which runs in $O(m(mn)^2)$ time for the hypergraph with m hyperedges and n nodes. In this section, we propose two heuristics for the MCHEC problem. Both of our heuristics not

Procedure HZRemoving
Input: A hypergraph on the same node set of the cycle.
Output: An embedding of the hypergraph in the cycle.
begin

1. Embed each hyperedge e_i as a c-path P_i' consisting of all its a-paths.
   The heaviest zone is defined to be the set of all the links in the cycle.
2. **For** each hyperedge e_i **do**
   delete the a-path that has the longest spanning in the heaviest zone from P_i'
   (If there is a tie, delete the longer one to break the tie.
   If still tied, arbitrarily delete one to break the tie);
   update the congestion on each link and set the new heaviest zone;

end.

Figure 4.5: Heaviest Zone Removing heuristic

only have a lower time complexity of $O(mn)$, but also beat the performance of the Iterated Maximum Independent Set heuristic according to our simulation results.

4.3.1 Heuristic 1: Heaviest Zone Removing

Recall that each hyperedge e_i with n_i nodes has exactly n_i adjacent paths (a-paths) in the cycle. Let P_i' denote a connecting path (c-path) for e_i in which all the n_i a-paths are present (i.e., P_i' is a circuit). Notice that any $n_i - 1$ a-paths of e_i form a feasible embedding (i.e., a c-path) for e_i. Therefore, we can construct a feasible embedding for each e_i by deleting exactly one a-path from P_i'. The Longest Adjacent Path Removing algorithm proposed by Lee and Ho [12] uses a greedy strategy that always deletes the longest a-path for each hyperedge. Thus, their strategy is actually based on the local information of each hyperedge.

To achieve better performance, our heuristic chooses one a-path to delete based on global information. We define the *heaviest zone* to be the set of links in the cycle with the maximum congestion. Our heuristic deletes the a-path that has the longest spanning in the heaviest zone for each hyperedge. The heuristic is given in Figure 4.5.

Figure 4.6 gives an example to illustrate the Heaviest Zone Removing heuristic. Initially, every hyperedge is embedded as a circuit and the heaviest zone is the whole cycle (see Figure 4.6(1)). We process hyperedges in an arbitrary order. For hyperedge $\{3, 5\}$, we delete the a-path from node 5 to node 3, which spans 4 links in the heaviest zone (see

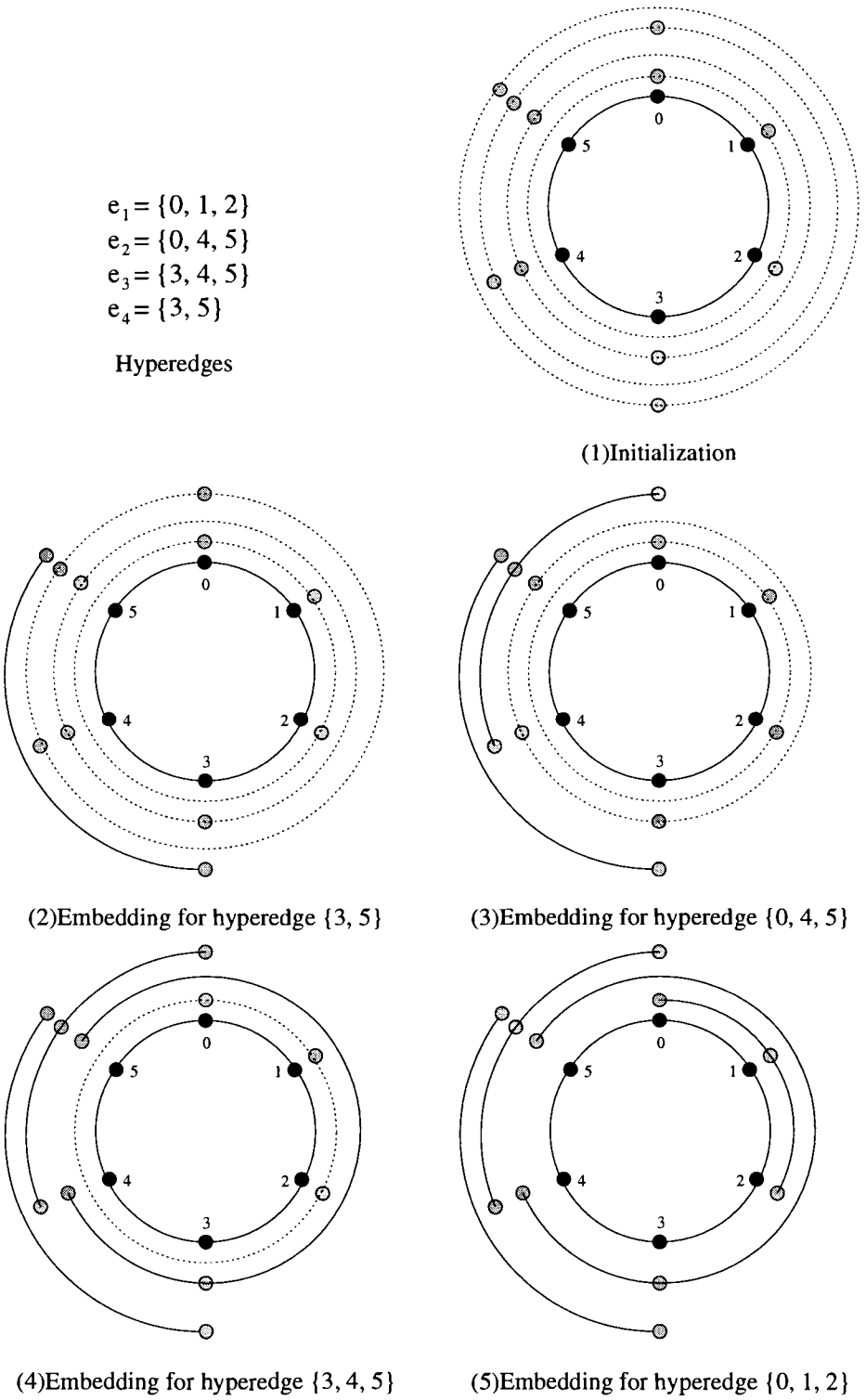

Figure 4.6: An example of Heaviest Zone Removing

Figure 4.6(2)). After updating the congestion of each link, the new heaviest zone consists of link 3 and link 4. For hyperedge $\{0, 4, 5\}$, both the a-path from node 4 to node 5 and the a-path from node 0 to node 4 have the longest spanning in the heaviest zone, but the a-path from node 0 to node 4 is longer, so we choose it to break the tie (see Figure 4.6(3)). After updating the congestion on each link, the new heaviest zone only includes link 4. We embed the other two hyperedges similarly. The embedding for all the hyperedges is given in Figure 4.6(5).

4.3.2 Heuristic 2: Spin Routing

The objective of the MCHEC problem is to minimize the maximum congestion over all the links in the cycle, so an intuitive idea is to evenly distribute congestions on all the links in the cycle. The following Spin Routing heuristic is developed based on this idea. Initially, Spin Routing chooses an arbitrary node as the start node (wlog, we assume that node 0 is the initial start node). An *embedding candidate* is defined to be a hyperedge that contains the start node. Assume that e_i is such an embedding candidate consisting of sorted nodes $\{v_1^i, v_2^i, \ldots, v_{n_i}^i\}$, and node v_j^i is the start node. The Spin Routing embeds e_i in the cycle as a c-path from v_j^i to v_{j-1}^i in the clockwise direction, and set node v_{j-1}^i to be the new start node. An embedding candidate that contains the new start point is re-embedded, and the start point is updated. The process is repeated until all the hyperedges are processed. The heuristic is given in Figure 4.7.

Figure 4.8 gives an example to illustrate the Spin Routing heuristic. Initially, node 0 is selected as the start node, and both hyperedges $\{0, 1, 2\}$ and $\{0, 4, 5\}$ are embedding candidates. However, if we embed them in the clockwise direction starting from start node 0, hyperedge $\{0, 1, 2\}$ has shorter spanning in the cycle. So we embed hyperedge $\{0, 1, 2\}$ as shown in Figure 4.8(1). The end point of the embedded path is node 2, which is set as the new start node. There is no un-embedded hyperedge containing start node 2, so we choose next node 3 in the cycle as the new start node, and then both hyperedges $\{3, 5\}$ and $\{3, 4, 5\}$ are embedding candidates. These two hyperedges have spanning with the same length as well, so we can choose any one to embed. Hyperedge $\{3, 4, 5\}$ is chosen as shown in Figure 4.8(2). The process is repeated until all the hyperedges have been embedded. The embedding for all the hyperedges is shown in Figure 4.8(4).

Procedure SpinRouting
Input: A hypergraph on the same node set of the cycle.
Output: An embedding of the hypergraph in the cycle.
begin
1. Without loss of generality, choose node 0 as the start node.
2. **While** there exist un-embedded hyperedges **do**
   if there exists the embedding candidate **then**
   embed the candidate as a c-path in the clockwise direction starting from the start node
   (If there is a tie, embed the one whose c-path has shorter spanning in the
   cycle to break the tie. If still tied, arbitrarily embed one to break the tie);
   set the end point of the newly embedded path as the start point.
   else
   set the node that is next to the current start node in the clockwise
   direction to be the start point.

end.

Figure 4.7: Spin Routing heuristic

4.4 Simulation results and conclusions

4.4.1 Simulation results

In this section, we present simulation results for the Clockwise Embedding [11], the Longest Adjacent Path Removing [12], our 1.8-approximation algorithm, the Iterated Maximum Independent Set heuristic [13], our Heaviest Zone Removing heuristic, and our Spin Routing heuristic. All the simulations use randomly generated hypergraphs. We assume that the size of each hyperedge is smaller than a pre-specified threshold. Below this threshold, a hyperedge can be in any size (≥ 2) with the same probability. This is based on the fact that each connection request usually involves only a few nodes in the network. We also assume that there is a pre-specified threshold for the total number of hyperedges in which a node is present. Below this threshold, a node can appear in any number of hyperedges with the same probability. This is based on the fact that a node can only be involved in a few connection requests simultaneously due to the node capacity.

We run simulations on both *sparse* and *dense* hypergraphs. In sparse hypergraphs, the number of hyperedges m is relatively small compared with the number of nodes n in the cycle. As shown in Table 4.1 and Table 4.3, the number of nodes n in the cycle is 100,

$e_1 = \{0, 1, 2\}$
$e_2 = \{0, 4, 5\}$
$e_3 = \{3, 4, 5\}$
$e_4 = \{3, 5\}$

Hyperedges

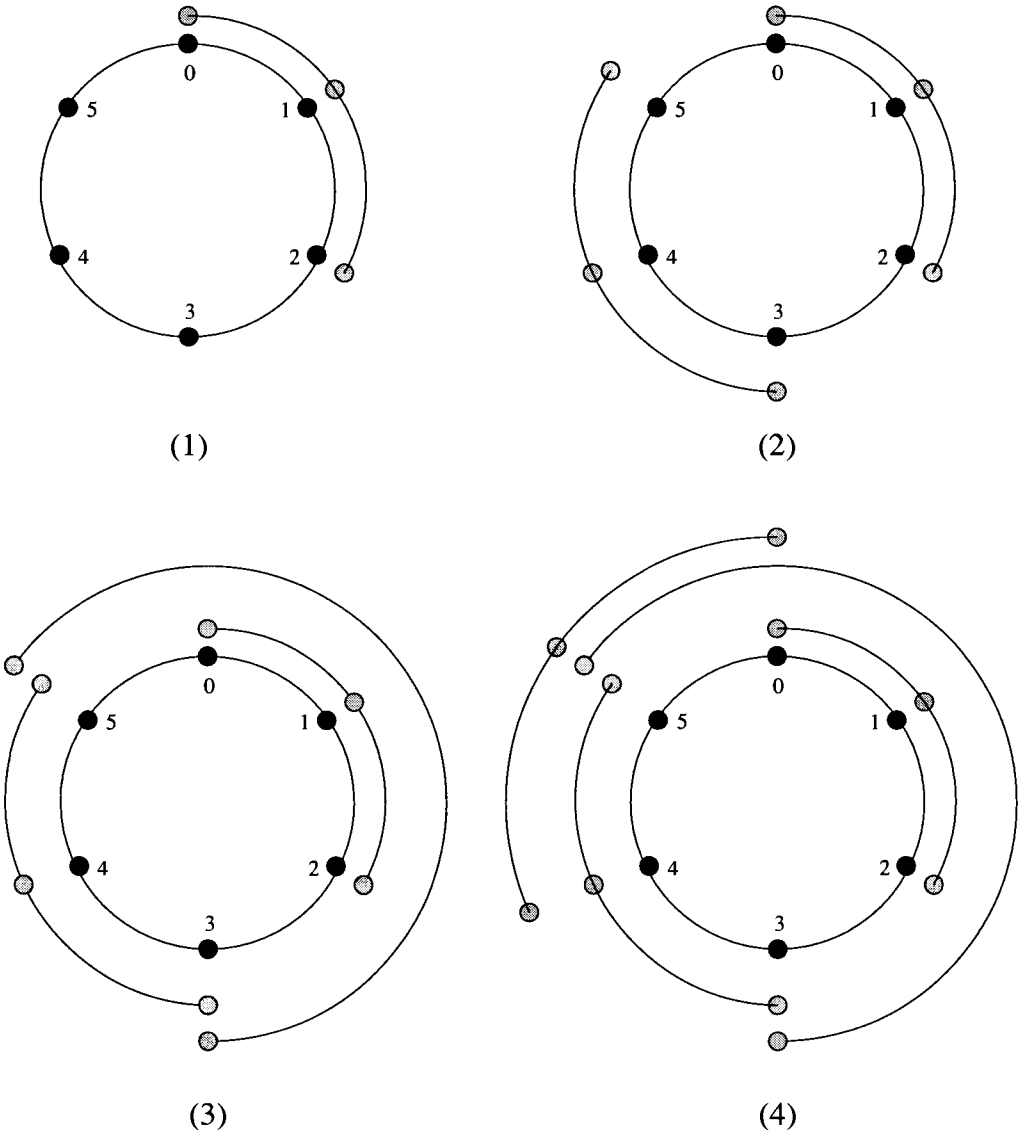

Figure 4.8: An example of Spin Routing

and the number of hyperedges m is in the range from $\frac{1}{20}n = 5$ up to $n = 100$. In dense hypergraphs, the number of hyperedges m is relatively large compared with the number of nodes n in the cycle. As shown in Table 4.2 and Table 4.4, the number of nodes n in the cycle is 30, and the number of hyperedges m is in the range from $n = 30$ up to $n^2 = 900$. We run simulations on cycles and hypergraphs with different sizes as well, and results from other simulations present similar characteristics. We only give four tables in this thesis.

m	LB	L	L_k	L_k/LB	$(L - L_k)/L$
5	3	4	4	1.333	0.000
10	7	8	8	1.142	0.000
15	9	15	12	1.333	0.200
20	12	19	17	1.416	0.105
25	15	25	21	1.400	0.160
30	18	29	26	1.444	0.103
35	20	33	28	1.400	0.151
40	23	39	35	1.521	0.102
45	27	44	40	1.481	0.090
50	27	44	39	1.444	0.113
55	31	50	42	1.354	0.160
60	34	58	49	1.441	0.155
65	38	60	51	1.342	0.150
70	38	62	55	1.447	0.112
75	43	72	62	1.441	0.138
80	45	75	65	1.444	0.133
85	46	77	69	1.500	0.103
90	51	84	76	1.490	0.095
95	56	93	81	1.446	0.129
100	56	98	86	1.535	0.122

Table 4.1: Practical performance of our 1.8-approximation algorithm on the cycle with $n = 100$ nodes

Table 4.1 shows the practical performance of our 1.8-approximation algorithm in terms of the maximum congestion. The cycle has $n = 100$ nodes. The first column m is the number of hyperedges, which is in the range from $\frac{1}{20}n = 5$ up to $n = 100$. The second column LB is a lower bound on the optimal maximum congestion. We compute this lower bound by choosing the larger one of the 2-link lower bound and 3-link lower bound. The third column L is the maximum congestion of the Clockwise Embedding. The fourth column L_k is the

m	LB	L	L_k	L_k/LB	$(L-L_k)/L$
30	18	29	25	1.388	0.137
60	36	58	52	1.444	0.103
90	52	85	78	1.500	0.082
120	67	115	102	1.522	0.113
150	82	140	122	1.487	0.128
180	101	171	154	1.524	0.099
210	117	198	170	1.452	0.141
240	134	225	200	1.492	0.111
270	144	252	217	1.506	0.138
300	167	283	245	1.467	0.134
330	182	311	266	1.461	0.144
360	197	331	292	1.482	0.117
390	210	356	315	1.500	0.115
420	226	392	337	1.491	0.140
450	246	420	367	1.491	0.126
480	263	451	399	1.517	0.115
510	275	463	405	1.472	0.125
540	300	504	443	1.476	0.121
570	315	534	472	1.498	0.116
600	326	555	490	1.503	0.117
630	343	581	511	1.489	0.120
660	356	615	539	1.514	0.123
690	371	634	556	1.498	0.123
720	393	671	592	1.506	0.117
750	402	685	604	1.502	0.118
780	426	735	645	1.514	0.122
810	436	759	664	1.522	0.125
840	453	773	679	1.498	0.121
870	467	804	710	1.520	0.116
900	480	818	714	1.487	0.127

Table 4.2: Practical performance of our 1.8-approximation algorithm on the cycle with $n = 30$ nodes

m	LB	L	L_k	L_{LAR}	L_{HZR}	L_{SR}	L_{IMIS}
5	3	4	4	3	3	3	3
10	7	8	8	7	7	8	8
15	9	15	12	10	9	12	13
20	12	19	17	14	13	17	18
25	15	25	21	18	16	20	21
30	18	29	26	24	19	23	26
35	20	33	28	22	21	25	29
40	23	39	35	27	24	31	34
45	27	44	40	29	28	35	38
50	27	44	39	29	28	38	40
55	31	50	42	35	32	39	43
60	34	58	49	39	36	44	50
65	38	60	51	39	39	44	52
70	38	62	55	40	39	47	54
75	43	72	62	51	44	54	64
80	45	75	65	51	48	58	67
85	46	77	69	50	50	61	70
90	51	84	76	56	56	64	78
95	56	93	81	63	58	68	82
100	56	98	86	61	60	71	82

Table 4.3: Performance comparison on the cycle with $n = 100$ nodes

m	LB	L	L_k	L_{LAR}	L_{HZR}	L_{SR}	L_{IMIS}
30	18	29	25	21	18	20	25
60	36	58	52	38	38	40	54
90	52	85	78	60	56	57	77
120	67	115	102	75	71	74	101
150	82	140	122	91	86	86	122
180	101	171	154	106	106	105	149
210	117	198	170	127	124	123	174
240	134	225	200	144	143	140	196
270	144	252	217	160	152	152	217
300	167	283	245	177	174	174	247
330	182	311	266	199	192	190	267
360	197	331	292	214	205	203	289
390	210	356	315	225	221	218	305
420	226	392	337	239	239	233	332
450	246	420	367	264	260	254	361
480	263	451	399	281	279	275	390
510	275	463	405	294	296	286	404
540	300	504	443	318	316	309	444
570	315	534	472	351	333	327	468
600	326	555	490	339	346	338	483
630	343	581	511	374	366	357	508
660	356	615	539	383	377	371	529
690	371	634	556	402	398	388	549
720	393	671	592	420	415	407	582
750	402	685	604	436	424	415	593
780	426	735	645	460	453	440	638
810	436	759	664	470	464	452	655
840	453	773	679	472	480	467	676
870	467	804	710	492	496	482	692
900	480	818	714	512	510	494	711

Table 4.4: Performance comparison on the cycle with $n = 30$ nodes

maximum congestion of our 1.8-approximation algorithm. The fifth column L_k/LB is the ratio to evaluate the practical performance of our 1.8-approximation algorithm. The last column $(L - L_k)/L$ gives the ratio to measure how much our 1.8-approximation algorithm decreases the maximum congestion of the Clockwise Embedding. In this simulation, the size of each hyperedge is restricted to the range from 2 to 7, and each node can be present in at most 7 hyperedges. Results show that the practical performance of our algorithm is even better than the guaranteed approximation ratio 1.8, and our algorithm decreases the maximum congestion of the Clockwise Embedding by about 10 percent in practice.

Table 4.2 shows the practical performance of our 1.8-approximation algorithm for the cycle with $n = 30$ nodes. The number of hyperedges m is in the range from $n = 30$ up to $n^2 = 900$. In this simulation, the size of each hyperedge is restricted to the range from 2 to 7 as well. Since the number of hyperedges is large, we assume that each node is present in at most $\frac{7m}{n}$ hyperedges, which guarantees that there are sufficient nodes to construct m hyperedges. Results show that the practical performance of our algorithm remains almost the same as that on the sparse hypergraph presented in Table 4.1, and our algorithm decreases the maximum congestion of the Clockwise Embedding by about 10 percent as well.

Table 4.3 shows a comparison of maximum congestions obtained by various algorithms. The cycle has $n = 100$ nodes. The number of hyperedges m is in the range from $\frac{1}{20}n = 5$ up to $n = 100$. The first four columns are the same as those in Table 4.1. The fifth, sixth, seventh and eighth columns are maximum congestion L_{LAR} of the Longest Adjacent Path Removing algorithm, L_{HZR} of our Heaviest Zone Removing heuristic, L_{SR} of our Spin Routing heuristic, and L_{IMIS} of the Iterated Maximum Independent Set heuristic respectively. In this simulation, the size of each hyperedge is restricted to the range from 2 to 7, and each node can be present in at most 7 hyperedges. Results show that our Heaviest Zone Removing heuristic has the best performance, which is very close to the lower bound. The maximum congestion of the Longest Adjacent Path Removing algorithm is slightly higher. Next is our Spin Routing heuristic. Our 1.8-approximation algorithm and the Iterated Maximum Independent Set heuristic are almost in the same level. Lastly, the Clockwise Embedding has the worst performance.

Table 4.4 shows a comparison of maximum congestions obtained by various algorithms for the cycle with $n = 30$ nodes. The number of hyperedges m is in the range from $n = 30$ up to $n^2 = 900$. In this simulation, the size of each hyperedge is restricted to the range from

2 to 7 as well. Since the number of hyperedges is large, we assume that each node is present in at most $\frac{7m}{n}$ hyperedges, which guarantees that there are sufficient nodes to construct m hyperedges. Results show that as the hypergraph becoming denser, our Spin Routing heuristic performs better than the Heaviest Zone Removing heuristic, and both of these two heuristics are slightly better than the Longest Adjacent Path Removing algorithm. The other algorithms remain the same performance ranks as those in Table 4.3.

4.4.2 Conclusions

In theory, our 1.8-approximation algorithm gives the best approximation ratio among all the approximation algorithms. It runs in optimal time complexity $O(mn)$ as well. In practice, our 1.8-approximation algorithm gives even better performance, and decreases the maximum congestion of the Clockwise Embedding by about 10 percent. The 2-approximation algorithm Longest Adjacent Path Removing performs better than our 1.8-approximation algorithm according to simulation results, however, there does exist the worst case in which the maximum congestion obtained by the Longest Adjacent Path Removing is exactly twice of the optimal maximum congestion.

Our two heuristics have very good practical performance, and also run in optimal time complexity $O(mn)$. For sparse hypergraphs, our Heaviest Zone Removing heuristic beats all the other algorithms. For dense hypergraphs, our Spin Routing heuristic has the best performance. The maximum congestion achieved by either heuristic is very close to the lower bound.

Chapter 5

Discussion and future work

In this thesis, we discussed routing problems on ring networks. Especially for the MCHEC problem, we proposed our own improved algorithms. In the future, we will continue our research in the following several aspects.

5.1 Further improvement on the algorithms for the MCHEC problem

In section 4, we gave a 1.8-approximation algorithm for the MCHEC problem. We proved a 3-link lower bound, which guarantees that our algorithm has a good approximation ratio even when the number of re-embedded hyperedges is small. The proof for the 3-link lower bound can be easily extended to obtain lower bounds involving more links. So a very straightforward idea for future research is to consider a 4-link lower bound, and then an open problem is how to design an algorithm to take advantage of the 4-link lower bound. Currently we are considering to use one of the links with the maximum congestion in the Clockwise Embedding, and the other three links used by the 3-link lower bound.

We also proposed an algorithm to compute optimal solutions for the MCHEC problem. Our algorithm runs in $O((\sqrt{mn})^{L^*})$ time. As we have mentioned in section 4, if more links are considered, there will be more constraints. Thus the number of checked embeddings, which dominates the time complexity of the algorithm, can be reduced further. So an open problem is to develop a more efficient polynomial time algorithm for subproblems with constant maximum congestions.

In addition, we intend to extend our results of the undirected model to the directed model. In the undirected model, each link in the cycle is used for bi-direction. Traffic can go back and forth along bidirectional links, and thus full connectivity is realized within each connection request. However, some networks such as optical networks do not support the technology of bidirectional use of links, and usually a pair of opposite-directed links are used between two adjacent nodes. So as an alternative, the directed model is proposed to study such networks. In the directed model, the cycle is considered as a symmetric digraph and each connection request is treated as an ordered node set. For unicast applications, each connection request consists of one source and one destination. A routing algorithm is required to find a directed path in the cycle from the source to the destination for each connection request. For more complicated communication applications such as multicast, each connection request consists of one source and several destinations. A routing algorithm is required to find a directed pseudo path in the cycle from the source to all the destinations, where the pseudo path for each connection request can be either a path in a single direction with the source as the start point of the path, or two joint paths in two opposite directions with the source as the start point of both paths. As to the MCHEC problem, since an embedding for the undirected model realizes full connectivity among nodes of each hyperedge, it is obvious that a solution for the undirected model provides a feasible embedding for the directed model as well. However, for every pair of opposite-directed links along each routing path in the cycle, only the link in either the clockwise or counterclockwise direction is used. This independent use of the two opposite-directed links provides more flexibility for designing routing algorithms. Trivially, any α-approximation($\alpha > 1$) algorithm for the undirected MCHEC problem is at least a 2α-approximation algorithm for the directed MCHEC problem, but this may not be a good approximation algorithm. How to design algorithms to achieve better approximation ratios is open. We will work on efficient algorithms for multicast and full connective routing requests among multiple nodes on directed cycles.

5.2 Minimum Congestion Weighted Hypergraph Embedding in a Cycle

We define the *size* of a connection request to be the amount of the bandwidth requirement of the connection request. A set of connection requests with non-uniform sizes can be described by a weighted hypergraph. The hyperedges with different weights in the weighted

hypergraph represent the connection requests with non-uniform sizes. Therefore, a more general case of the MCHEC problem, in which connection requests may have non-uniform sizes, is formulated by Lee and Ho [12] as the *Minimum Congestion Weighted Hypergraph Embedding in a Cycle* problem: Given a weighted hypergraph and a cycle on the same node set, embed the weighted hyperedges as weighted paths in the cycle such that the maximum congestion (i.e., the maximum total weight of paths using any single link in the cycle) over all the links in the cycle is minimized. A special case, in which each connection request involves exactly two nodes, can be formulated as the *Minimum Congestion Weighted Graph Embedding in a Cycle* problem.

5.2.1 Related work

1. A special case: Minimum Congestion Weighted Graph Embedding in a Cycle

   (a) NP-completeness

   The Minimum Congestion Weighted Graph Embedding in a Cycle problem is also well known as the *Ring Loading* problem [7, 14]. The Ring Loading problem is NP-complete, and various reductions from the *Partition* problem [21] have been constructed to prove its NP-completeness in [7, 14, 12].

   Theorem 5.2.1 *[7, 14, 12] The Ring Loading problem is NP-complete.*

   (b) Approximation algorithms

   i. 2-approximation algorithms

   Among the 2-approximation algorithms for the MCHEC problem, the Clockwise Embedding [11] and the Longest Adjacent Path Removing [12] can also be used to solve the Ring Loading problem with the same approximation ratio. In fact, the Clockwise Embedding belongs to a class of routing algorithms, which is called *Edge Avoidance Routing* by Cosares and Saniee [14]. In the Edge Avoidance Routing, all the connection requests are routed in the way to avoid containing a pre-specified link of the ring. For example, the highest numbered link $n-1$ in the ring is always avoided in the Clockwise Embedding.

   Furthermore, the Edge Avoidance Routing and the Longest Adjacent Path Removing belong to a more general class of routing algorithms, which is

called *Weight-based Routing* in [14]. The Weight-based Routing is defined as follows: Assign a non-negative weight for each link in the ring, and then route each connection request in the way that used links have minimum total weight. For example, the Edge Avoidance Routing is the Weight-based Routing in which the avoided link is assigned weight 1, and all the other links have weight 0; the Longest Adjacent Path Removing is the Weight-based Routing in which each link in the ring is assigned the same weight. A stronger result is proved in [14]: Any Weight-based Routing algorithm is a 2-approximation algorithm for the Ring Loading problem.

ii. An algorithm with solutions bounded above by $Optimum + \frac{3}{2}d_{max}$

Schrijver, Seymour, and Winkler [7] proposed an efficient algorithm with solutions exceeding the optimum by at most $\frac{3}{2}d_{max}$, where d_{max} is the size of the largest connection request. In their paper, Schrijver et al. considered a relaxed version of the Ring Loading problem in which every connection request can be split in an arbitrary way to route in both clockwise and counterclockwise directions. They solved the relaxed Ring Loading problem optimally, and then proved that after unsplitting the split connection requests by some strategies, the maximum congestion increase by at most $\frac{3}{2}d_{max}$. Notice that the optimal maximum congestion of the relaxed Ring Loading problem is a lower bound on the optimal maximum congestion of the Ring Loading problem, therefore, the maximum congestion obtained by their algorithm is bounded above by $Optimum + \frac{3}{2}d_{max}$.

iii. A polynomial time approximation scheme

Khanna [15] gave a polynomial time approximation scheme based on the above $Optimum + \frac{3}{2}d_{max}$ algorithm. The basic idea is that for any problem instance in which the size of the largest connection request is at most $\frac{2\epsilon \cdot Optimum}{3}$, a $(1 + \epsilon)$-approximation algorithm can be obtained using the approach in [7].

According to whether the size of a connection request is greater than $\frac{2\epsilon \cdot Optimum}{3}$, Khanna divided a set of connection requests into two disjoint subsets S_{high} and S_{low}. For the connection requests in set S_{low}, a routing with the maximum congestion bounded by $(1 + \epsilon) \cdot Optimum$ can be obtained by the algorithm in [7]. For the connection requests in set S_{high}, all the possible

routings are checked, and each possible routing is combined with the routing of set S_{low}. So a $(1+\epsilon)\cdot Optimum$ solution for the whole connection request set can be achieved eventually. For any fixed $\epsilon > 0$, Khanna's $(1+\epsilon)$-approximation algorithm runs in $O(\frac{1}{\epsilon}n^{\frac{3(1+1/n)}{\epsilon}+5})$ time.

(c) Variants of the Ring Loading problem

In the Ring Loading problem discussed above, it is assumed that no connection requests can be split. However, in some systems, a connection request can be split into two parts, with one part being routed in the clockwise direction and the other part being routed in the counterclockwise direction. If either of the two parts can have an arbitrary size, we refer to the Ring Loading problem as the *Ring Loading Problem With splitting (RLPW)*; if both of the two parts must have integral sizes, we refer to the Ring Loading problem as the *Ring Loading Problem With Integral splitting (RLPWI)*. The RLPW can be solved in polynomial time, but the RLPWI is proved to be NP-hard. Optimal algorithms for the RLPW and approximation algorithms for the RLPWI have been proposed in [7, 23, 24, 25, 26].

In addition, some research has also been done for the directed case of the Ring Loading problem. Wan and Yang [27] proved that the problem is NP-complete as well, and proposed several approximation algorithms to solve the directed Ring Loading problem.

2. Minimum Congestion Weighted Hypergraph Embedding in a Cycle

(a) NP-completeness

Since the MCHEC problem is a subproblem of the Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem, and the MCHEC problem is NP-complete, the following theorem holds:

Theorem 5.2.2 *[12] The Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem is NP-complete.*

(b) Approximation algorithms

Among the 2-approximation algorithms for the MCHEC problem, the Clockwise Embedding [11] and the Longest Adjacent Path Removing [12] can also be used to solve the Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem. Either of them has an approximation ratio of 2.

5.2.2 Future research

For the Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem, so far the best known approximation algorithms have the same approximation ratio of 2 [11, 12]. How to develop approximation algorithms with a better approximation ratio is an open problem.

As a special case of the Minimum Congestion Weighted Hypergraph Embedding in a Cycle problem, the Ring Loading problem has attracted much attention. The *Optimum* + $\frac{3}{2}d_{max}$ algorithm [7] is proper for instances in which d_{max} is not much bigger than the sizes of other connection requests, otherwise, the algorithm can't guarantee a good approximation ratio. The polynomial time approximation scheme [15] guarantees a good approximation ratio but runs with a high time complexity. In addition to these two results, all the other algorithms have approximation ratio at least 2. So a possible improvement is to develop algorithms running with relatively lower time complexities and better approximation ratios than 2.

5.3 Path Coloring problem

The *Path Coloring* problem is another important research issue. The problem is defined as follows: Given a set of connection requests, assign a path and a color to each connection request such that two connection requests receive different colors if their paths share a link in the network, and the objective is to minimize the number of used colors.

A popular application is the *Wavelength Division Multiplexing (WDM)* optical network, in which the available bandwidth on each optical fiber is partitioned into a certain number of channels, each at a different wavelength. In WDM optical networks, each wavelength can carry a separate stream of data. Due to the signal interference, any two paths that share an optical fiber must be assigned different wavelengths.

5.3.1 Related work

Previous research on the Path Coloring problem in ring networks can be categorized into the following directions.

1. Path Coloring with pre-specified routing

   Once routing is fixed, the Path Coloring problem in ring networks is actually the *Circular Arc Graph Coloring* problem, which has attracted much attention in the graph theory area. Garey et al. [28] showed that the problem is NP-hard. Tucker [29] gave an algorithm using at most $2L$ colors, where L is the maximum congestion on the ring. Tucker also conjectured that a Circular Arc Graph can be colored using at most $3\omega/2$ colors, where ω is the *Clique number* of the Circular Arc Graph. Hsu and Shih [30] gave an algorithm using at most $5\omega/3$ colors. Karapetian [31] proved Tucker's conjecture that a Circular Arc Graph can be colored using at most $3\omega/2$ colors. A randomized coloring algorithm that achieves approximation ratio $1 + 1/e + o(1)$ with a high probability was given by Kumar [32], where $e = 2.718$ is the base of natural logarithm. For the particular case of the *Proper Circular Arc Graph*, the coloring problem is solvable in polynomial time [33, 34, 35].

   There are also some work on the on-line version of the Circular Arc Graph Coloring. According to whether arrived paths can be released after some finite amount of time, there are incremental dynamic cases and purely dynamic cases. Slusarek [36] presented an algorithm that uses at most $3L - 2$ colors for the incremental case, where L is the maximum congestion on the ring. Kierstead and Trotter [37] showed that $3L - 2$ is also a lower bound for the incremental case. Gerstel et al. [38] studied the purely dynamic case, and presented an upper bound of $L + L \cdot \lceil \log_2 n \rceil$ and a lower bound of $0.5L \cdot (\lfloor \log_2 n \rfloor + 1)$, where n is the number of nodes in the ring.

2. Path Coloring without pre-specified routing

   In this case, the routing of connection requests is not fixed, and the objective is to route all the connection requests on the ring while using as few colors as possible.

   If all the connection requests have a uniform bandwidth requirement, the problem is called *Routing and Path Coloring* problem and proved to be NP-complete by Erlebach and Jansen [39]. Raghavan and Upfal [4] presented a 2-approximation algorithm for the undirected model. Mihail, Kaklamanis, and Rao [5] developed a 2-approximation algorithm for the directed model using a similar idea. It seems that nobody can beat the 2-approximation algorithm until Kumar [32] proposed a randomized approximation algorithm for the undirected model achieving approximation ratio $1.5+1/2e+o(1)$

with a high probability, where $e = 2.718$ is the base of natural logarithm. Cheng [40] presented an algorithm with approximation ratio $2 - \max\{4/n, 1/(50 \log n)\}$.

If connection requests have non-uniform bandwidth requirements, the problem is called *Demand Routing and Slotting Problem (DRSP)* and proved to be NP-complete by Carpenter et al. [41]. They also gave a 2-approximation algorithm for the DRSP by combining the Edge Avoidance Routing and the optimal *Interval Graph Coloring* algorithm presented by Golumbic [42]. Carpenter et al. also proved a stronger result that any Weight-based Routing followed by a basic coloring algorithm proposed by Tucker [29] is a 2-approximation algorithm for the DRSP.

3. Path Coloring with blocking allowed

   In the above two models, it is assumed that all the connection requests must be satisfied and the objective is to minimize the number of used colors. Some researchers considered the Path Coloring problem from a different view: Given a fixed number of colors, assign paths and colors to the connection requests to satisfy as many connection requests as possible, that is, the objective is to minimize the number of blocked connection requests. In a very recent paper, Christos, Aris, and Stathis [43] presented a $\frac{2}{3}$-approximation algorithm for the undirected model and a $\frac{7}{11}$-approximation algorithm for the directed model.

   Also, another research direction is to consider a random distribution of the connection requests and compute blocking probabilities [44, 45, 46].

4. Path Coloring with multi-fiber in WDM networks

   This model arises when a WDM optical network is allowed to use multiple parallel fibers. There are two approaches to study this problem. One is that the number of available wavelengths on each optical fiber is fixed, and the objective is to minimize the total number of used optical fibers in the network. This problem is formulated as the *Path Multicoloring* problem by Christos, Aris, and Stathis [47]. They presented algorithms for chains, rings, and stars. For ring networks, no matter whether routing is pre-specified or not, the problem is NP-complete. A 2-approximation algorithm was given in [47] for each case.

   The other approach is that the number of parallel optical fibers between any pair of

adjacent nodes is fixed, and the objective is to minimize the total number of used wavelengths. This problem is studied by Li and Simha [48], and Margara and Simon [49]. They proved independently that for pre-specified routing and a ring network with k parallel optical fibers, $\frac{k+1}{k} \cdot OPT$ is an upper bound on OPT, which denotes the optimal number of used wavelengths.

5. Path Coloring with wavelength conversion in WDM networks

   With the development of optical technologies, wavelength converters are deployed more and more widely in WDM optical networks. Using wavelength converters, we can significantly reduce the number of used wavelengths. However, the price of the wavelength converter is also an important factor for consideration. The previous work in this direction includes minimizing the number of wavelengths with full wavelength conversion or limited conversion, placing wavelength converters to make an optimal utilization, etc.

5.3.2 Future research

Although much work has been done, there are still many open problems for various Path Coloring problems in ring networks. Among all these problems, we think the following ones are worth investigating mostly.

First, the famous 2-approximation algorithms for the Routing and Path Coloring on ring networks proposed in [4, 5] have been mentioned by many researchers, and it seems difficult to beat their results. As pointed out by Cheng [40], once we can design a routing algorithm to minimize the total number of pairwise intersected paths optimally, then using the $\frac{3}{2}$-approximation algorithm proposed by Karapetian [31], we can achieve a $\frac{3}{2}$-approximation algorithm for the Routing and Path Coloring problem. But whether there exists such a polynomial time routing algorithm is still unknown. Second, the on-line version of the Routing and Path Coloring problem is also an interesting research direction. To our knowledge, the best approximation ratio is achieved by simply combining the Edge Avoidance Routing with the on-line Interval Graph Coloring algorithm presented in [41]. Third, there are also many open problems for further exploration in the multi-fiber model, wavelength conversion allowed model, and blocking allowed model.

Bibliography

[1] Thomas E. Stern and Krishna Bala. *Multiwavelength optical networks: a layered approach.* Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, 1999.

[2] Rajiv Ramaswami and Kumar N. Sivarajan. *Optical networks: a practical perspective.* 2nd Edtion. Morgan Kaufmann Publishers Inc.: San Francisco, CA, 2003.

[3] András Frank, Takao Nishizeki, Nobuji Saito, Hitoshi Suzuki, and Éva Tardos. Algorithms for routing around a rectangle. *Discrete Applied Mathematics*, 40:363–378, 1992.

[4] P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *Proc. 26th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 134–143, 1994.

[5] C. Kaklamanis M. Mihail and S. Rao. Efficient access to optical bandwidth. In *Proc. 36th Annual ACM Symposium on Foundations of Computer Science (FOCS)*, pages 548–557, 1995.

[6] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Perennes, and U. Vaccaro. Graph problems arising from wavelength-routing in all-optical networks. In *Proc. 2nd Workshop on Optics and Computer Science, part of IPPS*, 1997.

[7] Alexander Schrijver, Paul Seymour, and Peter Winkler. The ring loading problem. *SIAM J. Discrete Math.*, 11(1):1–14, 1998.

[8] G. Wilfong and P. Winkler. Ring routing and wavelength translation. In *Proc. 9th Annual ACM-SIAM Symp. on Discrete Algorithms.*

[9] Joseph L. Ganley and James P. Cohoon. Minimum-congestion hypergraph embedding in a cycle. *IEEE Trans. on Computers*, 46(5):600–602, 1997.

[10] Teofilo F. Gonzalez. Improved approximation algorithms for embedding hyperedges in a cycle. *Information Processing Letters*, 67:267–271, 1998.

[11] Tamra Carpenter, Steven Cosares, Joseph L. Ganley, and Iraj Saniee. A simple approximation algorithm for two problems in circuit design. *IEEE Trans. on Computers*, 47(11):1310–1312, 1998.

[12] SingLing Lee and Hann-Jang Ho. Algorithms and complexity for weighted hypergraph embedding in a cycle. In *Proc. 1st International Symposium on Cyber World (CW2002)*, 2002.

[13] Joseph L. Ganley and James P. Cohoon. A provably good moat routing algorithm. In *Proc. 6th Great Lakes Symposium on VLSI*, pages 86–91, 1996.

[14] Steve Cosares and Iraj Saniee. An optimization problem related to balancing loads on sonet rings. *Telecommunication Systems*, 3:165–181, 1994.

[15] Sanjeev Khanna. A polynomial time approximation scheme for the sonet ring loading problem. *Bell Labs Technical Journal*, pages 36–41, 1997.

[16] Teofilo F. Gonzalez and SingLing Lee. A 1.6 approximation algorithm for routing multiterminal nets. *SIAM J. on Computing*, 16:669–704, 1987.

[17] Teofilo F. Gonzalez and SingLing Lee. A linear time algorithm for optimal routing around a rectangle. *Journal of ACM*, 35(4):810–832, 1988.

[18] Andrea S. LaPaugh. A polynomial time algorithm for optimal routing around a rectangle. In *Proc. 21st Symposium on Foundations of Computer Science (FOCS80)*, pages 282–293, 1980.

[19] Majid Sarrafzadeh and F.P. Preparata. A bottom-up layout technique based on two-rectangle routing. *Integration: The VLSI Journal*, 5:231–246, 1987.

[20] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press: Cambridge, Mass. ; McGraw-Hill Book Company: Boston, 1990.

[21] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* W.H. Freeman: New York, 1979.

[22] U.I. Gupta, D.T. Lee, and J.Y.-T. Leung. Efficient algorithms for interval graphs and circular arc graphs. *Networks*, 12:459–467, 1982.

[23] Y. S. Myung, H. G. Kim, and D. W. Tcha. Optimal load balancing on sonet bidirectional ring. *Oper. Res.*, 45:148–152, 1997.

[24] C. Y. Lee and S. G. Chan. Balancing loads on sonet rings with integer demand splitting. *Comput. Oper. Res.*, 24:221–229, 1997.

[25] Rita Vachani, Alexander Shulman, and Peter Kubat. Multicommodity flows in ring networks. *INFORMS jouranl on Computing*, 8(3):235–242, 1996.

[26] Y. S. Myung. An efficient algorithm for the ring loading problem with integer demand splittin. *J. Discrete Math.*, 14(3):291–298, 2001.

[27] Peng-Jun Wan and Yuanyuan Yang. Load-balanced routing in counter rotated sonet rings. *Networks*, 35(4):279–286, 2000.

[28] M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal of Algebraic Discrete Methods*, pages 216–227, 1980.

[29] A. Tucker. Coloring a family of circular arcs. *SIAM Journal of Applied Mathematics*, 229(3):493–502, 1975.

[30] W. L. Hsu and W. K. Shih. An approximation algorithm for coloring circular arc graphs. In *SIAM Conference on Discrete Mathematics*, 1990.

[31] I. A. Karapetian. On coloring of arc graphs. *Dokladi of the Academy of Science of the Armenian Soviet Socialist Republic*, 70(5):306–311, 1980.

[32] V. Kumar. Approximating circular arc coloring and bandwidth allocation in all-optical ring networks. In *Proc. International Workshop on Approximation ALgorithms for Combinatorial Optimization*, pages 147–158. Springer-Verlag, 1998.

[33] J. B. Orlin, M. A. Bonuccelli, and D. P. Bovet. An $o(n^2)$ algorithm for coloring proper circular arc graphs. *SIAM J. Algebraic Discrete Methods*, 2(2):88–93, 1981.

[34] Wei kuan Shih and Wen-Lian Hsu. An $o(n^{1.5})$ algorithm to color proper circular arcs. *Discrete Applied Mathematics*, 25(3):321–323, 1989.

[35] A. Teng and Alan Tucker. An $o(qn)$ algorithm to q-coloring a proper family of circular arcs. *Discrete Mathematics*, 25:233–243, 1985.

[36] M. Slusarek. Optimal online coloring of circular arc graphs. *Inform. Theor. Appl.*, 29(5):423–429, 1995.

[37] H. Kierstead and W. Trotter. An extremal problem in recursive combinatorics. *Congressive Numerantium*, 33:143–153, 1981.

[38] Ori Gerstel, Galen Sasaki, Shay Kutten, and Rajiv Ramaswami. Worst-case analysis of dynamic wavelength allocation in optical networks. *IEEE/ACM Transactions on Networking*, 7(6):833–845, 1999.

[39] T. Erlebach and K. Jansen. Call scheduling in trees, rings and meshes. In *Proc. 30th Hawaii Int'l Conf. on System Science*, 1997.

[40] C. Cheng. A new approximation algorithm for the demand routing and slotting problem on rings with unit demands. *Lecture Notes in Computer Science 1671: Randomization, Approximation, and Combinatorial Optimization*, pages 209–220, 1999.

[41] T. Carpenter, S. Cosares, and I. Saniee. Demand routing and slotting on ring neworks. *Technical Report 97-02, Bellcore*, 1997.

[42] M. C. Golumbic. *ALgorithmic graph theory and perfect graphs.* Academic Press, Inc: San Diego, CA, 1980.

[43] Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Minimizing request blocking in all-optical rings. In *Proc. INFOCOM 2003.*

[44] R.A. Barry and P.A. Humblet. Models of blocking probability in all-optical networks with and without wavelength changers. *IEEE Journal on Selected Areas in Communications*, 14(5):858–867, 1996.

[45] G.N. Rouskas Y. Zhu and H.G. Perros. A path decomposition approach for computing blocking probabilities in wavelength routing networks. *IEEE/ACM Trans. Networking*, 8(6):747–762, 2000.

[46] G.N. Rouskas S. Ramesh and H.G. Perros. Computing blocking probabilities in multiclass wavelength routing networks with multicast calls. *IEEE Journal on Selected Areas in Communications*, 20(1):89–96, 2002.

[47] Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Routing and path multicoloring. *Information Processing Letters*, 80:249–256, 2001.

[48] G. Li and R. Simha. On the wavelength assignment problem in multifiber wdm star and ring networks. In *Proc. INFOCOM 2000.*

[49] L. Margara and J. Simon. Wavelength assignment problem on all-optical networks with k fibers per link. In *Proc. ICALP*, pages 768–779, 2000.

[50] Haruko Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31:75–81, 1981.

[51] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.*, 5:399–404, 1956.

[52] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5:97–101, 1958.

[53] T. C. Hu. Multi-commodity network flows. *Oper. Res.*, 11:344–360, 1963.

[54] Gordon Wilfong and Peter Winkler. Ring routing and wavelength translation. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 333–341, 1998.

[55] P.-J. Wan and L. Liu. Maximal throughput in wavelength-routed optical networks. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 46:15–26, 1998.