Efficient Algorithm for Embedding Hypergraphs in a Cycle

Qian-Ping Gu and Yong Wang

School of Computing Science
Simon Fraser University
Burnaby B.C. Canada V5A 1S6
Email: {qgu,ywangb}@cs.sfu.ca

Abstract: The problem of *Minimum Congestion Hypergraph Embedding in a Cycle (MCHEC)* is to embed the hyperedges of a hypergraph as paths in a cycle such that the maximum congestion (the maximum number of paths that use any single link in the cycle) is minimized. This problem has many applications, including minimizing communication congestions in computer networks and parallel computations. The MCHEC problem is NP-hard. We give a 1.8-approximation algorithm for the problem. This improves the previous 2-approximation results. The algorithm has the optimal $O(mn)$ time for the hypergraph with m hyperedges and n nodes.

Key words: Hypergraph embedding in a cycle, approximation algorithms, communication on rings, link congestion minimization.

Technical area: Algorithms

Corresponding author

Qian-Ping Gu

School of Computing Science

Simon Fraser University

Burnaby BC Canada V5A 1S6

Phone: 1-604-268-6705

Email: qgu@cs.sfu.ca

1 Introduction

The problem of *Minimum Congestion Hypergraph Embedding in a Cycle (MCHEC)* [3] is to embed the hyperedges of a hypergraph as paths in a cycle on the same number of nodes such that the maximum congestion is minimized, where the congestion of a link in the cycle is the number of paths that use the link. The MCHEC problem has applications in network communication, parallel computation, electronic design automation, and so on [2, 5, 6, 7, 9]. A communication application on a network can be defined by a set of routing requests and each request is defined by a subset of network nodes to be connected in the request. For one-to-one routing (unicast), a request has two nodes, the source and destination. For more complicated communication applications, such as multicast, a request may have more than two nodes of the network. In general, a communication application on a network can be further modeled as a hypergraph on the same node set of the network with each routing request in the application represented by a hyperedge in the hypergraph. To realize a communication application, a routing algorithm is required to set-up a virtual or dedicated routing path in the network to connect the nodes in each hyperedge. A most important issue in designing the routing algorithm is to minimize the congestion on the links in the network. An algorithm for the MCHEC problem realizes the communication application represented by the hypergraph on the ring with link congestion minimized. Similarly, in parallel computation, the processors of a parallel computer can be represented by the nodes of the hypergraph and a group of processors which are required to communicate with each other can be represented by a hyperedge. A solution to the MCHEC problem gives a routing for the communications among the processors in each group on the ring connected parallel computers with the congestion on links minimized.

If each hyperedge of the hypergraph contains exactly two nodes then the MCHEC problem becomes *graph embedding in a cycle* problem and the optimal solution of the problem can be solved in polynomial time [2]. For general hypergraphs (a hyperedge can contain more than two nodes), Ganley and Cohoon proved that the MCHEC problem is NP-hard and gave a 3-approximation algorithm for the problem [3]. They also gave an algorithm which determines if an instance of the MCHEC problem has a solution with maximum congestion l in $O((mn)^{l+1})$ time in [3] for hypergraphs with m hyperedges and n nodes. The $O((mn)^{l+1})$ result implies that the MCHEC problem can be solved in polynomial time if the maximum congestion is a constant. For non-constant maximum congestions, better approximation algorithms have been proposed based on different approaches [1, 4, 8]. In [1] a simple algorithm based on *clockwise embedding* is proposed for the MCHEC problem. In clockwise embedding, the nodes in the cycle are clockwise numbered by $0, 1, ..., n-1$ and the nodes in a hyperedge are connected by a path in the cycle from the smallest node to the largest one in the clockwise direction. In [4] two algorithms were given for the MCHEC problem. The first algorithm transforms the MCHEC problem to an Integer Linear Programming (ILP) problem. Approximate solutions for the ILP problem are obtained from rounding-off the solutions of the corresponding linear programming problem in polynomial time. The second algorithm transforms the MCHEC problem to the graph embedding in a cycle problem. In [8] an algorithm based on *longest adjacent path removing* was given. All the algorithms in [1, 4, 8] have the approximation ratio two.

In this paper, we give a 1.8-approximation algorithm for the MCHEC problem. Our algorithm starts from the clockwise embedding. Then the algorithm tries to re-embed some hyperedges to reduce the maximum congestion. Let L be the maximum congestion in the clockwise embedding and L^* be the maximum congestion in the optimal embedding. Our algorithm tries to re-embed k hyperedges to get an embedding of maximum congestion $L-k$ for k as large as possible. The approximation ratio of the algorithm is $(L-k)/L^*$. Since $\lceil L/2 \rceil \leq L^*$ [1], the approximation ratio of our algorithm is at most $(L-k)/\lceil L/2 \rceil$. This gives a good approximation ratio for large k. For the case that k is small, we prove a new lower bound on L^*. Roughly speaking, the approximation ratio of the algorithm increases from 1 to 1.8 when k decreases from $L/2$ to $L/10$, the ratio decreases from 1.8 to 1.5 when 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.

The rest of the paper is organized as follows. In Section 2, we give the preliminary of the paper and review the clockwise embedding. Section 3 gives our algorithm and the analysis of the algorithm. The final section concludes the paper.

2 Preliminary

A *cycle* C of n nodes is an undirected graph with node set $\{i|0 \leq i \leq n-1\}$. 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 modulo n operation. A *hypergraph* $H(V, E_H)$ of n nodes and m hyperedges is a hypergraph with node set $V = \{i|0 \leq i \leq n-1\}$ and hyperedge set $E_H = \{e_1, ..., e_m\}$, where each hyperedge e_i is a subset of V with two or more nodes. 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 nodes in e_i are in p_i. An embedding of hypergraph $H(V, E_H)$ in cycle C is a set of c-paths in C that each hyperedge of $H(V, E_H)$ has exactly one c-path in the set. Given an embedding of a hypergraph, the congestion of each link of C is the number of c-paths that contain the link. The MCHEC problem is that given a hypergraph and a cycle on the same node set, find an embedding of the hypergraph such that the maximum congestion of any link in the cycle is minimized.

We assume that the nodes of C are clockwise numbered by $0, 1, ..., n-1$ and link $(i, i+1)$ in C is numbered i (see Figure 1). In [1], a simple algorithm called clockwise embedding is given: The c-path for each hyperedge e_i in the given hypergraph connects all the nodes of e_i in C from the smallest node to the largest node in the clockwise direction. Figure 1 gives the embedding of the hypergraph with edges $e_1 = \{0, 1, 8\}, e_2 = \{1, 4, 7\}$, $e_3 = \{2, 3, 4\}$, and $e_4 = \{3, 5, 7\}$.

A *segment* of C is a connected subgraph of C. Cycle C is cut into two segments by removing any two links. We call the two links a *cut* of C. A hyperedge is *separated* by a cut if there is a node of the hyperedge in each of the two segments obtained from the cut. Obviously, the c-path in any embedding for a separated hyperedge must contain at least one of the two links in the cut. Let $N(x, y)$ be the number of hyperedges separated by links x and y. Then $\lceil N(x, y)/2 \rceil \leq L^*$. For the clockwise embedding, define $l(i)$ to be the congestion of link i in C and $L = \max\{l(i)|0 \leq i \leq n-1\}$ be the maximum congestion. Notice that $l(n-1) = 0$. Assume that link s has the maximum congestion L. Then $N(n-1, s) = L$. In any embedding, the c-apth for a separated hyperedge must contain at least one of the links $n-1$ and s. From this, we have $\lceil L/2 \rceil \leq L^*$ and the approximation ratio of the clockwise

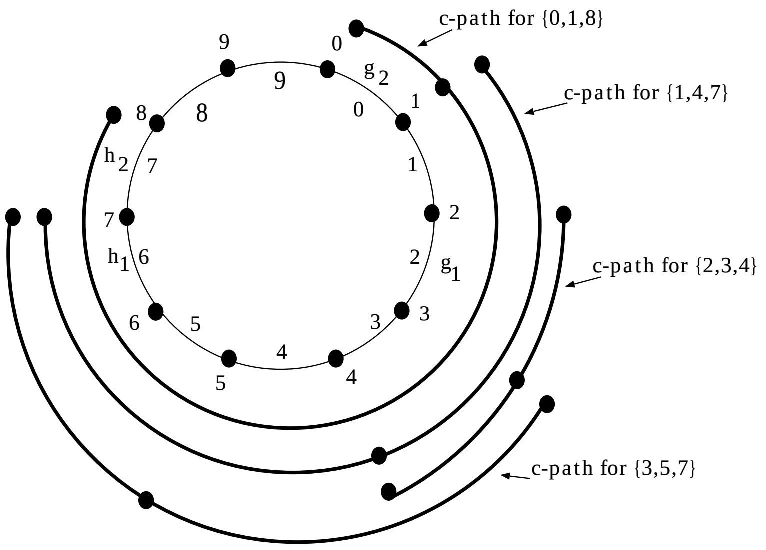

Figure 1: The cycle and clockwise embedding

embedding is $L/L^* \leq L/\lceil L/2 \rceil \leq 2$ [1].

Define *segment* $\langle i, j \rangle$ of C to be the segment of C which includes the nodes from i to j in clockwise direction.

3 Embedding algorithm

In this section, we describe and analyze our algorithm for the MCHEC problem. We first introduce some terminologies for this purpose. Recall that $l(i)$ is the congestion of link i and L is the maximum congestion in the clockwise embedding, and L^* is the maximum congestion in the optimal embedding. For integer k with $1 \leq k \leq \lfloor L/2 \rfloor$, let g_k be the smallest link with $l(g_k) \geq L - 2k + 1$. Similarly, let h_k be the largest link with $l(h_k) \geq L - 2k + 1$. Notice that $0 \leq g_k \leq h_k < n - 1$. In Figure 1, $L = 4$, $g_k = 0$ and $h_k = 7$ for $k = 2$. For $k = 1$, $g_k = 2$ and $h_k = 6$. 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 $\langle g_k + 1, h_k \rangle$. In Figure 1, hyperedge $\{0, 1, 8\}$ is a candidate w.r.t. $k = 1$. For each candidate w.r.t. k, we can embed it in such a way that the c-path does not contain any link i with $g_k \leq i \leq h_k$ (see Figure 2). From the definitions of g_k, h_k, and the

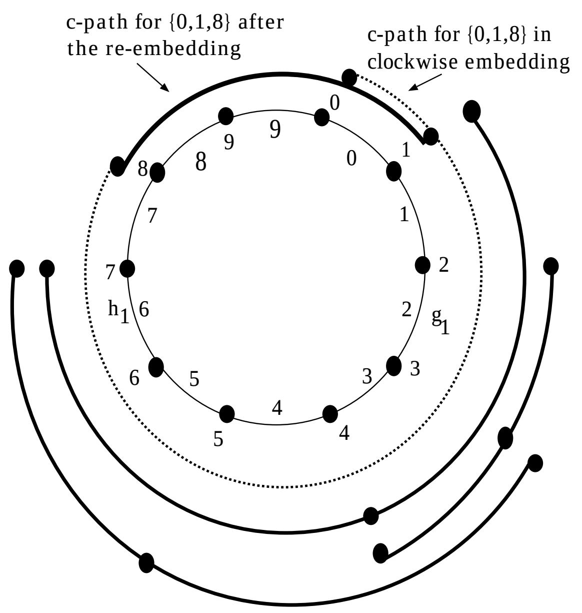
c-path for {0,1,8} after the re-embedding
c-path for {0,1,8} in clockwise embedding
0
1
2
3
4
5
6
7
8
9
h1
g1

Figure 2: The c-paths for the candidate w.r.t. $k = 1$ in clockwise embedding and after the re-embedding.

candidate w.r.t. k, we can get $g_{k+1} \leq g_k$, $h_{k+1} \geq h_k$, and a candidate w.r.t. $k + 1$ is also a candidate w.r.t. k (the inverse may not be true). Let x_k be the number of candidates w.r.t. k. Then $x_{k+1} \leq x_k$.

The outline of our algorithm is as follows: We start with the clockwise embedding. Then we try to re-embed the candidates w.r.t. k such that the c-path for each candidate does not contain any link i with $g_k \leq i \leq h_k$. 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 outlined above has approximation ratio 1.8 except for a few special cases of constant L. Although the $O((mn)^{L^*+1})$ time algorithm in [3] can be used to find the optimal embedding for constant L, the time complexity could be too high in practice. We give a subroutine to handle those special cases. The subroutine is efficient and guarantees the 1.8 approximation ratio for the special cases. Our algorithm and subroutine are given in Figure 3.

Algorithm R_Embedding terminates with k taking one of $\{0, 1, ..., \lfloor L/2 \rfloor\}$. Let L_k be the maximum congestion of the embedding by the algorithm.

```
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.
    2. Find links $g_k$ and $h_k$ for $k = 1, 2, ..., \lfloor L/2 \rfloor$.
       $x_k$ is defined 0 for $k = \lfloor L/2 \rfloor + 1$ and $k = 0$.
    3. For $k := \lfloor L/2 \rfloor$ step $-1$ until 1 do
           if ($x_k \geq k$) then goto step 4.
    4. If $x_k \geq k + 1$ and $x_{k+1} \geq 1$ then
           re-embed $k + 1$ 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 $k$ 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 the link with the 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 $e_i$ by $p_i$ reduces the maximum congestion by one then
                re-embed $e_i$ by $p_i$ and return
    else /* $L = 12$ and $k = 1$ */
        for every pair $e_i, e_j \in E$ do
            if re-embedding $e_i$ by $p_i$ and $e_j$ by $p_j$ reduces the maximum congestion by two then
                re-embed $e_i$ by $p_i$ and $e_j$ by $p_j$ and return
        re-embed one candidate w.r.t. $k = 1$.
    Return
end.
```

Figure 3: The embedding algorithm for the MCHEC problem.

Lemma 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. Since for link i with $g_k \leq i \leq h_k$, $l(i) \leq L$, and for i with $i < g_k$ or $i > h_k$, $l(i) \leq L - 2k$, we can have an embedding of maximum congestion $L - k$.

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

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 $L - 1$. $\square$

From the above lemma, the approximation ratio of our algorithm can be obtained from $(L-k)/L^*$ or $(L-k-1)/L^*$. For large k, we can use $\lceil L/2 \rceil$ as the lower bound on L^* and get the approximation ratio $(L-k)/\lceil L/2 \rceil$ or $(L-k-1)/\lceil L/2 \rceil$ for our algorithm. This suggests that when the algorithm terminates with a large k, we have a good approximation ratio. However, if the algorithm terminates with a small k, e.g., $k = 0$ then the approximation ratio given by $L_k/\lceil L/2 \rceil$ is 2, no better than the clockwise embedding. In what follows, we prove that if the algorithm terminates with a small k then a lower bound better than $\lceil L/2 \rceil$ on L^* can be found. By this better lower bound, we can get the 1.8 approximation ratio of our algorithm. The lower bound $\lceil L/2 \rceil$ is obtained from the cut of two links in the cycle. The new lower bound derived below involves three links $n-1$, g_k, and h_k.

To derive the lower bound, we need some new notation. For arbitrary links g and h in the cycle with $0 \leq g < h < n-1$, we define

W: the set of the hyperedges such that each hyperedge has a node in segment $\langle 0, g\rangle$, a node in segment $\langle g+1, h\rangle$, and a node in segment $\langle h+1, n-1\rangle$;

X: the set of the hyperedges such that each hyperedge has a node in segment $\langle 0, g\rangle$, has NO node in segment $\langle g+1, h\rangle$, and has a node in segment $\langle h+1, n-1\rangle$;

Y: the set of the hyperedges such that each hyperedge has a node in segment $\langle 0, g\rangle$, a node in segment $\langle g+1, h\rangle$, and has NO node in segment $\langle h+1, n-1\rangle$; and

Z: the set of the hyperedges such that each hyperedge has NO node in segment $\langle 0, g\rangle$, has a node in segment $\langle g+1, h\rangle$, and a node in segment $\langle h+1, n-1\rangle$.

In Figure 1, let $g = 2$ and $h = 6$. Then hyperedge $\{1, 4, 7\}$ is in W, hyperedge $\{0, 1, 8\}$ is in X, hyperedge $\{2, 3, 4\}$ is in Y, and hyperedge $\{3, 5, 7\}$ is in Z.

The intuition for proving the new lower bound on L^* is as follows: When Algorithm R_Embedding terminates with a small k, the number of c-paths in clockwise embedding that use link g_k (and h_k) is close to L. A hyperedge whose c-path in clockwise embedding uses link g_k (resp. h_k) belongs to one of the sets W, X, or Y (resp. W, X or Z). A key observation is that a c-path in any embedding for a hyperedge in W must contain at least two of the three links $n-1$, g_k, and h_k. This implies that the lower bound on L^* is at least $2|W|/3$. Also, a hyperedge in X is separated by links $n-1$ and g_k (or h_k). Similarly, a hyperedge in Y (resp. Z) is separated by links g_k and h_k (or $n-1$) (resp. by links h_k and $n-1$ (or g_k)). Based on the above observations, a better lower bound than $\lceil L/2\rceil$ on L^* can be obtained for small k. Especially we have the following result.

Lemma 2 *For any links g and h with $0 \leq g < h < n-1$,*

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

Proof: Let I be an arbitrary embedding. For any links g and h with $0 \leq g < h < n-1$, define

$l_I(i)$: the number of c-paths in embedding I that contain link i for the hyperedges of $W \cup X \cup Y \cup Z$;

$\bar{w}_I(i)$: the number of c-paths in embedding I that DO NOT contain link i for the hyperedges of W;

$x_I(i)$: the number of c-paths in embedding I that contain link i for the hyperedges of X;

$y_I(i)$: the number of c-paths in embedding I that contain link i for the hyperedges of Y; and

$z_I(i)$: the number of c-paths in embedding I that contain link i for the hyperedges of Z.

Then

$$L^* \geq \min_I \{\max\{l_I(n-1), l_I(g), l_I(h)\}\}.$$

The number of c-paths in embedding I for the hyperedges in W that contain link $n-1$ is $|W| - \bar{w}_I(n-1)$. The number of c-paths in I for the hyperedges in X that contain link $n-1$ is $x_I(n-1)$. For a hyperedge in Y, if the c-path for the hyperedge does not contain link g then the c-path must contain links $n-1$ and h. From this, the number of c-paths in I that contain link $n-1$ for hyperedges in Y is $|Y| - y_I(g)$. Similarly, the number of c-paths in I that contain link $n-1$ for hyperedges in Z is $|Z| - z_I(h)$. Summarizing the above,

$$l_I(n-1) \geq (|W| - \bar{w}_I(n-1)) + x_I(n-1) + (|Y| - y_I(g)) + (|Z| - z_I(h)).$$

By a similar argument, we have

$$l_I(g) \geq (|W| - \bar{w}_I(g)) + (|X| - x_I(n-1)) + y_I(g) + (|Z| - z_I(h))$$

and

$$l_I(h) \geq (|W| - \bar{w}_I(h)) + (|X| - x_I(n-1)) + (|Y| - y_I(g)) + z_I(h).$$

Then

$$\begin{aligned} l_I(n-1) + l_I(g) + l_I(h) \quad \geq \quad & 3|W| - (\bar{w}_I(n-1) + \bar{w}_I(g) + \bar{w}_I(h)) \\ & +2(|X| + |Y| + |Z|) - (x_I(n-1) + y_I(g) + z_I(h)). \end{aligned}$$

Obviously, $x_I(n-1) \leq |X|$, $y_I(g) \leq |Y|$, and $z_I(h) \leq |Z|$. For any hyperedge in W, the c-path for the hyperedge contains at least two of the three links $n-1$, g, and h. Therefore,

$$(\bar{w}_I(n-1) + \bar{w}_I(g) + \bar{w}_I(h)) \leq |W|.$$

Thus, we have

$$l_I(n-1) + l_I(g) + l_I(h) \geq 2|W| + |X| + |Y| + |Z|. \tag{1}$$

To prove the lemma by contradiction, assume that

$$l_I(n-1) < \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|),$$

$$l_I(g) < \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|),$$

and

$$l_I(h) < \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

Then

$$l_I(n-1) + l_I(g) + l_I(h) < 2|W| + |X| + |Y| + |Z|,$$

a contradiction to inequality (1). Thus, for any embedding I,

$$\max\{l_I(n-1), l_I(g), l_I(h)\} \geq \frac{2}{3}|W| + \frac{1}{3}(|X| + |Y| + |Z|).$$

From this, the lemma is proved. $\square$

Notice that there are hypergraphs with L^* tight to the lower bound in Lemma 2. An example of such hypergraphs is as follows: Let g and h be two arbitrary links with $0 \leq g < h < n-1$ 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 $\langle h+1, 0\rangle$ (resp. $\langle 0, g+1\rangle$, $\langle g+1, h+1\rangle$). The hyperedges in W consist of the hyperedges in three disjoint subsets W_{n-1}, W_g, and W_h. Each hyperedge in W_{n-1} (resp. W_g, W_h) does not contain any node in segment $\langle h+2, n-1\rangle$ (resp. $\langle 1, g\rangle$, $\langle g+2, h\rangle$). The cardinalities of W_{n-1}, W_g, and W_h are defined by

$$\begin{aligned} |W_{n-1}| &= \frac{|W| + 2|X| - |Y| - |Z|}{3}, \\ |W_g| &= \frac{|W| - |X| + 2|Y| - |Z|}{3}, \text{ and} \\ |W_h| &= \frac{|W| - |X| - |Y| + 2|Z|}{3}. \end{aligned}$$

Let I be an embedding for $H(V, E_h)$ such that the c-path for a hyperedge in W_i does not contain link i for $i = n-1, g, h$, the c-path for a hyperedge in X (resp. Y, Z) does not contain

g or h (resp. h or $n-1$, $n-1$ or g). Then for any link i with $h+1 \le i \le n-1$,

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

Similarly, for any link i with $0 \le i \le g$,

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

and for any link i with $g+1 \le i \le h$,

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

Now, we use the lower bound given in Lemma 2 to derive the approximation ratio for our algorithm for small k.

Theorem 3 *The approximation ratio of Algorithm R_Embedding is bounded by 1.8.*

Proof: Assume that Algorithm R_Embedding terminates with the maximum congestion L_k. From Lemma 2,

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

for any links g and h with $0 \le g < h < n-1$. Since $|Y| = l(g) - |W| - |X|$ and $|Z| = l(h) - |W| - |X|$, we have

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

Notice that for links g_k and h_k, if $g_k = h_k$ then g_k is the unique link with the maximum congestion L. From the definition of x_k, we have $x_k = L$ for $g_k = h_k$. 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 (2), we have $|X| = x_{k+1}$. Since the algorithm terminates with maximum congestion L_k, we have $x_{k+1} \le k$. This implies $g_{k+1} < h_{k+1}$. Since $l(g_{k+1}) \ge L - 2(k+1) + 1$ and $l(h_{k+1}) \ge L - 2(k+1) + 1$,

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

From this, 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^*,

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

Therefore, the approximation ratio of the algorithm is bounded 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 $k \geq \lfloor L/10 \rfloor$, $(L-k-1)/\lceil L/2 \rceil \leq 1.8$, and 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 (2), then $|X| = x_k = k$ and $g_k < h_k$. Since $l(g_k) \geq L-2k+1$ and $l(h_k) \geq L-2k+1$,

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

From this, $L^* \geq \lceil L/2 \rceil$, and $L_k = L-k$, the approximation ratio of the algorithm is bounded by

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

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

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

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

The approximation ratio of the algorithm is bounded by

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

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

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

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

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

To complete the proof of the theorem, we need to show that $L_k/L^* \leq 1.8$ for $\langle 1\rangle$, $\langle 2\rangle$, and $\langle 3\rangle$. We only show the most complex case of $\langle 3\rangle$ here due to the space limit. The proofs for the other cases are similar and will be given in the full version of the paper.

Recall that s is the link with the maximum congestion $L = 12$ 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 for $e_i \in E$ that does not contain link s. There are two cases for $L = 12$ and $k = 1$. Case (1) is that re-embedding some $e_i, e_j \in E$ by p_i, p_j reduces the maximum congestion by two. In this case, from $L = 12$ we have $L_k = 10$, implying $L_k/\lceil L/2\rceil = 10/6 \leq 1.8$.

Case (2) is that re-embedding any two hyperedges of E does not reduce the maximum congestion by two. In this case, one candidate w.r.t. $k = 1$ is re-embedded in Subroutine Special_Cases and $L_k = 11$. Let I be an embedding with the optimal maximum congestion L^*. We prove that $L^* \geq 7$ which implies $L_k/L^* \leq 11/7 \leq 1.8$. The proof is partitioned in several subcases.

Notice that every hyperedge in E is separated by the cut of links s and $n-1$. So, the c-path in any embedding for $e_i \in E$ must contain at least one of the links s and $n-1$. If the c-path in I for one hyperedge in E contains both links s and $n-1$, then from $|E| = 12$ the sum of the congestions of s and $n-1$ is at least 13 which implies $L^* \geq 7$. Assume that the c-path in I for a hyperedge in E contains exactly one of the links s and $n-1$. If more than six c-paths for the hyperedges of E contain link s (or link $n-1$) then $L^* \geq 7$.

We assume that six c-paths contains link s and the other six c-paths contains link $n-1$. Notice that for any hyperedge $e \notin E$, the c-path $p(e)$ for e in the clockwise embedding contains neither s nor $n-1$ and any c-path for e other than $p(e)$ contains both s and $n-1$. So, in the embedding I, if any hyperedge $e \notin E$ is embedded with a c-path other than $p(e)$ then $L^* \geq 7$. So, we can assume that every hyperedge $e \notin E$ is embedded by clockwise embedding in I. By this assumption, embedding I is obtained from re-embedding six $e_i's \in E$ by $p_i's$. Let $E' \subset E$ be the set of those six hyperedges. To show $L^* \geq 7$ by contradiction, assume that $L^* = 6$. Then, re-embedding any k of the six hyperedges $e_i's \in E'$ by $p_i's$ will reduce the maximum congestion by k. This is true especially for $k = 2$, a contradiction to the condition for Case (2). Thus, $L^* \geq 7$. $\square$

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. So, the time complexity of Algorithm R_Embedding is $O(mn)$ which is optimal since the total number of links in the c-paths for any embedding for the hypergraph can be $\Omega(mn)$.

4 Concluding remarks

A 1.8-approximation algorithm is given for the MCHEC problem. This improves the previous 2-approximation results. The improvement is based on the following observations: If some hyperedges can be re-embedded such that the maximum congestion of the clockwise embedding can be reduced then we have a better embedding. Otherwise, a better lower bound on L^* can be obtained to guarantee the approximation ratio. Whether the 1.8-approximation ratio can be improved further is open. A possible approach is to find a better lower bound on L^* by looking at four links of the cycle (the lower bound of this paper is obtained by considering three links). Find better approximation algorithms for specific classes of hypergraphs may be worth further investigation as well.

Acknowledgment

This work was partially supported by the NSERC research grant (RGPIN250304), President Research Grant and Endowed Research Fellowship of Simon Fraser University.

References

[1] T. Carpenter, S. Cosares, J.L. Ganley, and I. Saniee. A simple approximation algorithm for two problems in circuit design. *IEEE Trans. on Computers*, 47(11):1310–1312, 1998.

[2] A. Frank, T. Nishizeki, N. Saito, and H. Suzuki E. Tardos. Algorithms for routing around a rectangle. *Discrete Applied Mathematics*, 40:363–378, 1992.

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

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

[5] T. Gonzalez and S.L. Lee. A 1.6 approximation algorithm for routing multiterminal nets. *SIAM J. on Computing*, 16:669–704, 1987.

[6] T. Gonzalez and S.L. Lee. A linear time algorithm for optimal routing around a rectangle. *Journal of ACM*, 35(4):810–832, 1988.

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

[8] S.L. Lee and H.J. Ho. Algorithms and complexity for weighted hypergraph embedding in a cycle. In *Proc. of the 1st International Symposium on Cyber World (CW2002)*, page To Appear, 2002.

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