Improved Approximation Algorithms for Embedding Hypergraphs in a Cycle

Teofilo F. Gonzalez
Department of Computer Science
University of California,
Santa Barbara, CA, 93106
teo@cs.ucsb.edu
UCSB TR 97-23
Revised: June 98

June 29, 1998

Abstract

Approximation algorithms embedding hypergraphs in a cycle so as to minimize the maximum congestion are presented. Our algorithms generate an embedding by transforming the problem into another problem that can be solved in polynomial time. One transforms it to a Linear Programming (LP) problem, and the other one (LP-Free) to the problem of embedding a graph in a cycle. Both algorithms generate an embedding with congestion at most twice of that in an optimal solution, and we give problem instances for which the solutions generated by both of these algorithms is about twice of optimal. Our problem has applications in CAD and parallel computation.
Keywords: Approximation Algorithms, parallel computation, CAD, linear time algorithms, minimize congestion.

1 Introduction

The problem of Embedding Hypergraphs in a Cycle so a to Minimize Congestion (EHCMC) [3] has applications in design automation (routing nets around a rectangle ([5] [6], [7], [10]), and moat routing([1], [4], [8], [12])), and in parallel computing (mapping data structures onto a ring network [3]). The EHCMC problem consists of a hypergraph $H = (V, E_H)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of n vertices, and $E_H = \{h_1, h_2, \ldots, h_m\}$ is a set of m hyperedges, i.e., each hyperedge is a subset of two or more vertices. The cycle C defined over the set of vertices V consists of the edges $E_C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \text{ and } \{v_n, v_1\}\}$. A *connecting path or (simply a c-path)* P_i in C, for $1 \le i \le m$, is a path in C such that all the vertices in the i^{th} hyperedge are in P_i. Our problem consists of finding c-paths P_i, for $1 \le i \le m$, such that the congestion which is measured by $\mathbf{max}_{e \in C} |\{i \text{ such that } e \in P_i\}|$ is least possible.

When the hypergraph is a graph (all hyperedges contain exactly two vertices), Frank, Nishizeki, Saito, Suzuki, and Tardos [2] developed a linear time algorithm to solve this problem. Ganley and Cohoon [3] showed that the problem is NP-hard in general, and that it can be solved in polynomial time when there is an embedding with congestion at most k, for any fixed value k. Ganley and Cohoon [3] also developed an approximation algorithm that generates embedding with congestion at most three times the congestion of an optimal solution, i.e., $\hat{f}_I \leq 3f_I^*$, where I is an instance of the EHCMC problem, $\hat{f}_I$ is the congestion of the solution generated by the algorithm for instance I, and f_I^* is the congestion of an optimal solution to I. The time complexity of the algorithm is linear with respect to the sum of the cardinalities of the hyperedges. In this paper we present two new approximation algorithms that generate solutions whose congestion is within two times the congestion of an optimal solution. The two algorithms are similar in nature. The difference is that the first algorithm transforms the problem to a linear programming problem, and the other transforms it to the embedding a graph in a cycle problem. The latter algorithm takes linear time with respect to the total number of vertices in the hyperedges, and in general it provides solutions with smaller congestion. For all $n \geq 3$ and odd, we give problem instances such that both of the algorithms generate a solution with congestion $\hat{f}_I = (2 - \frac{3}{n+1})f_I^*$.

2 Linear Programming Approximation Approximation

Let m_i be the number of vertices in hyperedge h_i, an let $h_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,m_i})$ be the ordered set of vertices in h_i, i.e., $h_{i,1} \leq h_{i,2} \leq \ldots \leq h_{i,m_i}$. We define the *adjacent paths (or simply a-paths)* $p_{i,j}$, for $1 \leq j \leq m_i$, as the set of edges in C from vertex $h_{i,j}$ to vertex $h_{i,j+1}$, where h_{i,m_i+1} is defined as $h_{i,1}$.

A feasible solution (or simply a solution) to the EHCMC problem is a set of c-paths, one for each hyperedge. For hyperedge h_i a c-path P_i consist of at least $m_i - 1$ of its a-paths. Note that if the c-path P_i consists of all its m_i a-paths, then one of the a-paths may be deleted without affecting feasibility.

For each a-path $p_{i,j}$ we define the variable $a_{i,j}$ which holds a real value in the interval $[0, 1]$. We define the inequalities

$$\begin{aligned} \sum_{j=1}^{m_i} a_{i,j} = m_i - 1 & \qquad 0 \leq i \leq m \\ 0 \leq a_{i,j} \leq 1. & \qquad \forall\ i \text{ and } j \end{aligned} \tag{1}$$

Any solution to (1) must satisfy that at least $m_i - 1$ of the $a_{i,\cdot}$'s have value at least $1/2$. We define the *round function* $r(x)$ as 1 if $x \geq 1/2$ and 0 otherwise. Applying the round function to any feasible solution to (1) (i.e., $r(a_{i,j})$) we know that for each value of i at least $m_i - 1$ of the rounded values $r(a_{i,\cdot})$ are equal to one, and at most one of the $r(a_{i,\cdot})$ has the value of zero. For each i and a solution to (1) we define the c-path Q_i as a-path $p_{i,j}$ is in Q_i iff $r(a_{i,j}) = 1$. The paths Q_is are a feasible solution to the EHCMC problem. Clearly, every solution to the EHCMC can be generated from a solution to (1) by following this rounding procedure.

The decision version of the EHCMC problem is to find an embedding into C with congestion at most k, i.e., the total number of c-paths covering each edge in C is at most k. This additional constraint can be modeled by the following equations.

For each edge e in the cycle C add the equation $\displaystyle\sum_{p_{i,j} \text{ includes } e} a_{i,j} \le k.$

The EHCMC problem is to find the least k such that there is an embedding into C with congestion at most k. Let us now consider the following LP program.

$$
\begin{array}{lrl}
\text{Find Minimum } k \text{ such that} & & \\
& \displaystyle\sum_{p_{i,j} \text{ includes } e} a_{i,j} \le k & \forall\, e \in C \\
& \displaystyle\sum_{j=1}^{m_i} a_{i,j} = m_i - 1 & 0 \le i \le m \\
& 0 \le a_{i,j} \le 1 & \forall\, i \text{ and } j
\end{array}
\tag{2}
$$

Any feasible solution to any instance I of the EHCMC is a feasible solution to (2). Therefore, $f_I^* \ge k^*$, where k^* is the objective function value (k) of an optimal solution to the LP program (2) corresponding to instance I. Now consider an optimal solution to (2) for problem instance I. Clearly the rounded values $r(a_{i,j})$ satisfy the same properties as the rounded values for (1), but we cannot claim that the congestion of the embedding constructed from the rounded solution to (2) is at most k^* because the rounding increases congestion. However, the congestion is at most doubled, i.e., it is at most $2k^*$. Therefore, the embedding generated for any instance I of the EHCMC problem from the rounded solution to an optimal solution to the LP program (2) for instance I is such that $\hat{f}_I \le 2f_I^*$.

Lemma 2.1 *The embedding generated for any instance I of the EHCMC problem from the rounded solution to an optimal solution to the LP program (2) for instance I satisfies $\hat{f}_I \le 2f_I^*$. The time complexity of this procedure is bounded by a polynomial on the number of bits that represent the instance of the EHCMC problem.*

Proof: By the above discussion and the facts that the currently fastest algorithm to solve LP problems is polynomial on the number of input bits, and the fact that all the numbers in the LP instance are integers between 0 and n, remember that n is the number of vertices.

□

Let us now consider the following example.

Example 2.1 *The problem instance is depicted in Figure 1. Each vertex in the graph C belongs to exactly one hyperedge* [1].

The a-paths for each of the three hyperedges are given below. Note that $m_1 = 2$ and $m_2 = m_3 = 3$.

[1]Note that in general a vertex may belong to several hyperedges.

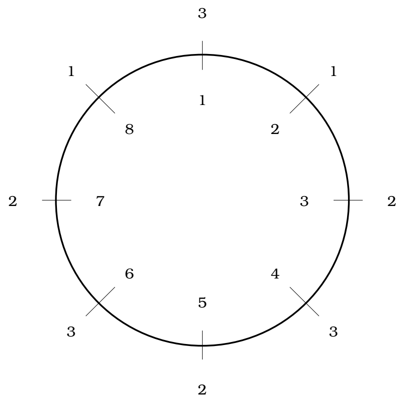

Figure 1: The vertex numbers are inside the circle and the numbers outside indicate the index of the hyperedge they belong.

$$
\begin{aligned}
p_{1,1} &= \{(2,3),(3,4),(4,5),(5,6),(6,7),(7,8)\} \\
p_{1,2} &= \{(8,1),(1,2)\} \\
p_{2,1} &= \{(3,4),(4,5)\} \\
p_{2,2} &= \{(5,6),(6,7)\} \\
p_{2,3} &= \{(7,8),(8,1),(1,2),(2,3)\} \\
p_{3,1} &= \{(1,2),(2,3),(3,4)\} \\
p_{3,2} &= \{(4,5),(5,6)\} \\
p_{3,3} &= \{(6,7),(7,8),(8,1)\}
\end{aligned}
$$

The LP program for this problem instance is given below.

Find Min k such that

$$
\begin{aligned}
a_{1,2} + a_{2,3} + a_{3,1} &\le k && \text{for } e = (1,2) \\
a_{1,1} + a_{2,3} + a_{3,1} &\le k && \text{for } e = (2,3) \\
a_{1,1} + a_{2,1} + a_{3,1} &\le k && \text{for } e = (3,4) \\
a_{1,1} + a_{2,1} + a_{3,2} &\le k && \text{for } e = (4,5) \\
a_{1,1} + a_{2,2} + a_{3,2} &\le k && \text{for } e = (5,6) \\
a_{1,1} + a_{2,2} + a_{3,3} &\le k && \text{for } e = (6,7) \\
a_{1,1} + a_{2,3} + a_{3,3} &\le k && \text{for } e = (7,8) \\
a_{1,2} + a_{2,3} + a_{3,3} &\le k && \text{for } e = (8,1) \\
a_{1,1} + a_{1,2} &= 1 \\
a_{2,1} + a_{2,2} + a_{2,3} &= 2 \\
a_{3,1} + a_{3,2} + a_{3,3} &= 2 \\
0 \le a_{i,j} &\le 1 && \forall\ i \text{ and } j
\end{aligned}
$$

An alternate LP formulation is defined over the complement of the variables $a_{i,j}$, which we call $\bar{a}_{i,j}$. By the complement of a variable we mean $a_{i,j} = 1 - \bar{a}_{i,j}$. For this case our

LP formulation is simpler because the last set of inequalities is not needed since LP assumes nonnegative real values for its variables.

Let us now establish the tightness of the approximation bound, i.e., find problem instances for which the approximation bound is tight or almost tight. For the algorithms in this section the approximation bound is asymptotic to 2. In what follows we give problem instances I_n for $n \geq 3$ and odd such that the solution generated by the algorithm has congestion $2n-1$ (the optimal solution to the LP problem has objective function value n), but the optimal congestion is $n+1$. Thus the approximation bound for I_n is $2 - \frac{3}{n+1}$.

For $n = 3$, problem instance I_3 is given in Figure 2(A). An optimal solution with congestion 3 (rather than $n+1 = 4$) is given in Figure 2(B). An LP optimal solution is given in Figure 2(C) (the dotted lines means that the variable corresponding to the a-path has value $\frac{1}{2}$; and the thick line means that it has value 1; otherwise it is zero)[2]. Therefore, the objective function value of an optimal solution to the LP problem is 3. The solution to the EHCMC problem generated by our LP method has congestion 5, and the approximation is $2 - \frac{1}{n} = \frac{5}{3}$, which is slightly larger that $2 - \frac{3}{n+1}$.

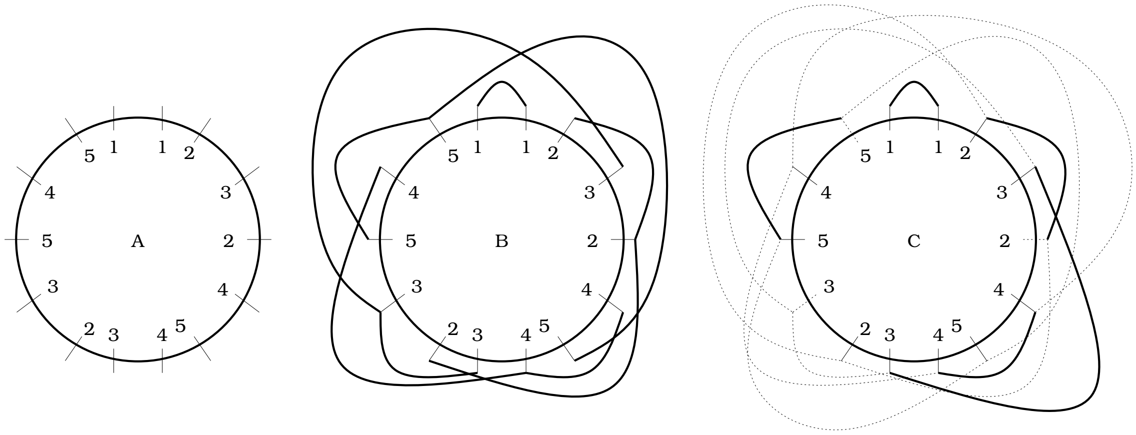

Figure 2: The hyperedge points (indices) are inside the circle.

For $n > 3$ and odd, the problem instance I_n is defined as follows. There are $2n-1$ hyperedges that we refer to as hyperedges $0, 1, ..., n-1, 1', 2', ..., (n-1)'$. All the hyperedges have three vertices, except for hyperedge 0 that has only 2 vertices. The sequence of hyperedge points around the circle is as follows:

$$0, 0, r_0, h_1, r_1, h_2, r_2, \ldots, h_{n-2}, r_{n-2}, h_{n-1}, r_{n-1}, r^{'}_{n-1}, h^{'}_{n-1}, r^{'}_{n-2}, h^{'}_{n-2}, \ldots, r^{'}_2, h^{'}_2, r^{'}_1, h^{'}_1, r^{'}_0,$$

where the r_is and h_is are sequences of hyperedge points defined as follows:

Table 1: The r_i and h_i sequences.

h_1	h_2	h_3	$\ldots$	h_i	$\ldots$	h_{n-2}	h_{n-1}
$\emptyset$	$((n-1)', (n-2)')$	$(n-3)'$	$\ldots$	$(n-i)'$	$\ldots$	$2'$	$1'$
$h^{'}_{n-1}$	$h^{'}_{n-2}$	$\ldots$	$h^{'}_i$	$\ldots$	$h^{'}_3$	$h^{'}_2$	$h^{'}_1$
1	2	$\ldots$	$n-i$	$\ldots$	$n-3$	$(n-2, n-1)$	$\emptyset$

[2]Actually this is the solution generated in less than one second by the simplex method on a Pentium 133 MHz PC.

r_0	r_1	r_2	...	r_i	...	r_{n-3}	r_{n-2}	r_{n-1}
1	$(2,1)$	$(3,2)$	...	$(i+1,i)$	...	$(n-2,n-3)$	$(n-1,n-2)$	n-1

r'_{n-1}	r'_{n-2}	r'_{n-3}	...	r'_i	...	r'_2	r'_1	r'_0
(n-1)'	$((n-2)',(n-1)')$	$((n-3)',(n-2)')$	...	$((i)',(i+1)')$	...	$(2',3')$	$(1',2')$	$1'$

The case when $n = 5$ is given below.

$$0, 0, r_0, h_1, r_1, h_2, r_2, h_3, r_3, h_4, r_4, r'_4, h'_4, r'_3, h'_3, r'_2, h'_2, r'_1, h'_1, r'_0,$$

where the r_is and h_is are sequences of hyperedge points defined as follows:

Table 2: The r_i and h_i sequences for $n = 5$.

h_1	h_2	h_3	h_4	h'_4	h'_3	h'_2	h'_1
$\emptyset$	$(4',3')$	$2'$	$1'$	1	2	$(3,4)$	$\emptyset$

r_0	r_1	r_2	r_3	r_4	r'_4	r'_3	r'_2	r'_1	r'_0
1	$(2,1)$	$(3,2)$	$(4,3)$	4	$4'$	$(3',4')$	$(2',3')$	$(1',2')$	$1'$

The problem instance is thus

$$\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{1}, 4', 3', \mathbf{3}, \mathbf{2}, 2', \mathbf{4}, \mathbf{3}, 1', \mathbf{4}, \mathbf{4'}, 1, \mathbf{3'}, \mathbf{4'}, 2, \mathbf{2'}, \mathbf{3'}, 3, 4, \mathbf{1'}, \mathbf{2'}, \mathbf{1'}$$

An optimal solution to the LP problem has a value of one for all the variables representing a-paths between pairs of adjacent hyperedge points in bold; a zero for the variables representing the remaining a-path for hyperedge zero, and a value $\frac{1}{2}$ for the variables representing the remaining a-paths of the hyperedges[3]. The objective function value of an optimal solution is therefore 5, but the solution to the EHCMC problem generated from it has congestion 9. So the approximation bound is $2 - \frac{1}{n}$ when we take as the optimal solution value the objective function value of an optimal solution to the LP problem. However, the best possible solution to the EHCMC problem has a congestion equal to 6 rather than 5. This solution connects hyperedge 0 in the obvious way; hyperedges 1,2,3 and 4 are connected by paths that start at the leftmost point of the hyperedge continuing to the right, and terminating at the rightmost point of the hyperedge; and hyperedges $1'$,$2'$,$3'$, and $4'$ are connected by a path that does not include the a-path from the leftmost point to the next point to the right it. It is simple to verify that the congestion for this solution is 6.

We claim that the general problem instance I_n has an LP optimal solution with objective function value n, and the solution to the EHCMC problem that we generate from it has congestion $2n - 1$, but the best possible solution to the EHCMC problem has congestion $n + 1$. Thus, the ratio between the congestion of the solution generated by our algorithm and the best possible solution is $2 - \frac{3}{n+1}$, which gets as close as possible to 2 as n grows without actually reaching the bound 2.

[3] Actually this is the solution generated by the simplex method for $n \leq 39$. For $n > 39$ the system runs out of space, and for $n = 39$ a Pentium 133 MHz PC took about 1.75 minutes to generate this solution.

3 LP-Free Approximation Algorithm

The main problem with the LP formulation is the time required to solve the LP problem. Even though the time is polynomial with respect to the number of bits in the input, the constant associated with the time complexity bound is huge. Solving the LP problem via the simplex method takes, in the worst case, exponential time, but, on average, it is fast. To guarantee that we can always generate solutions very quickly, we develop an alternate method with the same worst case approximation bound, but the time required to find it is linear with respect to the input size. Furthermore, the constant associated with the time complexity bound is small.

To simplity the exposition we assume that each vertex in V belongs to exactly one hyperedge. Later on we show how to solve the general case. We solve this problem by transforming it to an instance of the EHCMC problem in which all the hyperedges have exactly two vertices which we know can be solved in linear time by the algorithm in [2]. Then from the solution to that problem we solve our EHCMC problem. The transformation is as follows: Each hyperedge h_i with m_i points is transformed into m_i edges (i.e., hyperedges with two vertices). For example the problem instance in Example 1 is transformed to the instance given in Figure 3. Note that hyperedge 1 in Figure 1 becomes edges (1,1) and (1,2) in Figure 3(A), and edges 1 and 2 in 3(B). This is represented by $1 \rightarrow ((1,1),(1,2)) \rightarrow 1,2$. Similarly $2 \rightarrow ((2,1),(2,2),(2,3)) \rightarrow 3,4,5$ and $3 \rightarrow ((3,1),(3,2),(3,3)) \rightarrow 6,7,8$.

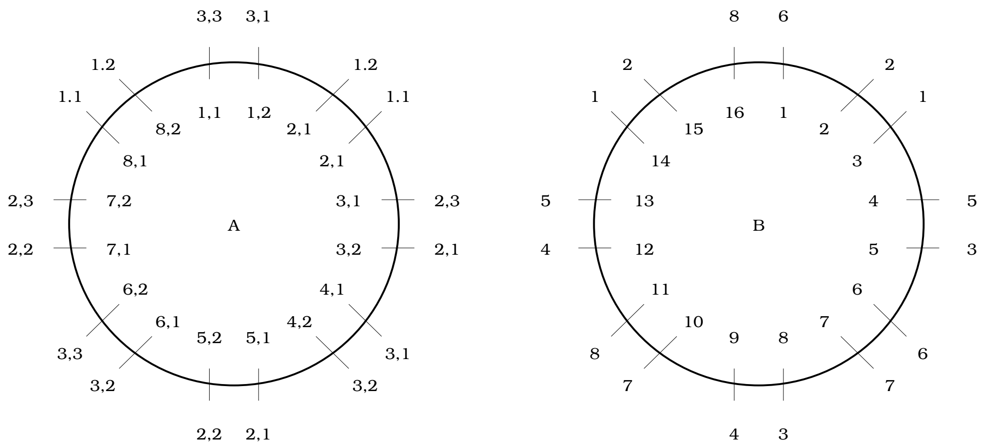

Figure 3: The vertex numbers are inside the circle and the numbers outside indicate the hyperedge to which they belong.

A hyperedge with three points is transformed into three edges and there are many ways the three edges could be interconnected. However, any interconnection pattern can be transformed into the patterns given in Figure 4 (A, B, C and D) without increasing the congestion. For example patterns E and F in Figure 4 can be transformed into the one given in Figure 4(A) without increasing the congestion. Once we have constructed the instance of the EHCMC problem in which all the hyperedges have exactly two vertices we solve it with the linear time algorithm given in [2]. Now we transform back such solution to the original problem and remove duplicate connections, or delete an a-path from a c-path that includes all its a-paths. The solution has congestion $\hat{f}_I \leq 2f_I^*$ simply because an optimal solution to the original problem can be transformed into a feasible solution to the transformed problem with congestion $2f_I^*$ and the algorithm in [2] gives the best possible solution to the transformed problem.

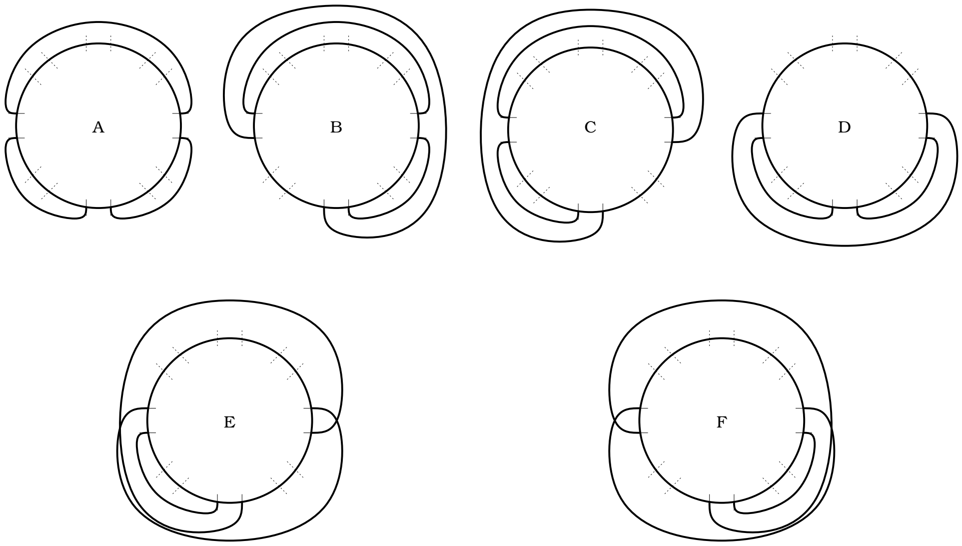

Figure 4: Possible paths for the transformed problem of a hyperedge with three points.

Lemma 3.1 *The procedure described above generates for every instance of the EHCMC a solution with congestion $\hat{f}_I \leq 2f_I^*$. The time complexity of our procedure is linear with respect to the total cardinality of the hyperedges.*

Proof: By the above discussion.

□

For $n > 3$ and odd, let I_n be the problem instance defined in the previous section. We now prove that the solution generated by the method given in this section has congestion equal to $2 - \frac{3}{n+1}$ times the optimal congestion value. Lets define as the *marked points* in I_n the points between the two adjacent vertices in hyperedge zero, and the point between r_{n-1} and r'_{n-1}. As in the previous section we know that $f^*_{I_n} = n+1$. Now let us consider a solution generated by the algorithm in this section. No matter how the hyperedges in I_n are connected, their paths must cross at least two times one of the marked points or once both of the marked points. Since there are $2n-1$ hyperedges, then the total connections over the marked points is $4n-2$. Therefore, $\hat{f}_{I_n} \geq 2n-1$. Now the solution that connects all hyperedges as in Figure 4(A) is a possible output of the algorithm. So even if one removes duplicate paths, the solution generated by the method in this section has congestion equal to $(2 - \frac{3}{n+1})f^*_{I_n}$.

4 Discussion

A slightly better approach results by applying the last transformation, but leaving out for each hyperedge one of the transformed edges. Perhaps the best choice for the edge to be deleted is random. For hyperedges with four points there is a simpler construction that uses only two edges that join the opposite points of the hyperedge. We should note that the above heuristics do not improve the worst case apporximation bound, but, in general, a better solution may be obtained. There are other iterative procedures and heuristics one can employ. In practice one could run several variations of our algorithms and then find and output the best of these solutions. However, so far we have not been able to find an approximation algorithm whose worst case approximation bound is smaller than two.

We have assumed that each vertex in C is in exactly one hyperedge. In general, a vertex may be in more than one hyperedge. The LP algorithm adapts to this problem without any major change, and the LP-free algorithm can also handle this case because the algorithm for embedding a graph into a cycle [2] also works for this type of problems.

References

[1] Baker, B. S., and R. Y. Pinter "An Algorithm for the Optimal Placement and Routing of a Circuit within a Ring of PADS," IEEE FOCS, pp. 360 – 370, Oct. 1983.

[2] Frank, A., T. Nishizeki, N. Saito, H. Suzuki, and E. Tardos, "Algorithms for Routing Around a Rectangle," *Discrete Applied Mathematics,* Vol. 40, No. 3, 1992, pp. 363 – 378.

[3] Ganley, J. L., and J. P. Cohoon, "Minimum-Congestion Hypergraph Embedding in a Cycle," *IEEE Transactions on Computers,* Vol. 46, No. 5, pp. 600 – 602, May 1997.

[4] Ganley, J. L., and J. P. Cohoon, "A Provably Good Moat Routing Algorithm," Proceedings of The Sixth Great Lakes Symposium on VLSI, pp. 86 – 91, 1996.

[5] Gonzalez, T., and S. L. Lee, "A Linear Time Algorithm for Optimal Wiring Around a Rectangle," *JACM,* Vol. 35, No. 4, pp. 810 - 832, Oct. 1988.

[6] Gonzalez, T., and S. L. Lee, "A 1.60 Approximation Algorithm for Routing Multiterminal Nets Around a Rectangle," *SIAM J. Computing,* Vol. 16, No. 4, pp. 669 - 704, 1987.

[7] LaPaugh, A. S., "A Polynomial Time Algorithm for Optimal Routing Around a Rectangle", Proceedings of the 21st IEEE FOCS, pp. 282 – 293, 1980.

[8] McGehee, R. K., "A practical Moat Router," Proceedings of The Twenty-Fourth Design Automation Conference, pp. 216 – 222, 1987.

[9] Okamura, H. and P. D. Seymour, "Multicommodity Flows in Planar Graphs," *Journal of Combinatorial Theory, Series B,* Vol. 31, pp. 75 – 81, 1981.

[10] Sarrafzadeh M. and F. P. Preparata, "A Bottom-Up Layout Technique Based on Two-Rectangle Routing," *INTEGRATION: the VLSI journal*, Vol. 5, pp. 231 – 246, 1987.

[11] Suzuki, H., T, Nishizeki, and N. Saito, "Variable Priority Queue and Doughnut Routing," *Journal of Algorithms,* Vol 13, pp. 606 – 635, 1992.

[12] Tsai, C. C., and Chen, S. J., "A Linear Time Algorithm for Planar Moat Routing," *Journal of Information Science and Engineering,* Vol. 10, No. 1, pp. 111 – 128, 1994.