ELSEVIER

Information Processing Letters 67 (1998) 267–271

Information Processing Letters

Improved approximation algorithms for embedding hyperedges in a cycle

Teofilo F. Gonzalez [1]

Department of Computer Science, University of California, Santa Barbara, CA 93106, USA

Received 1 December 1997
Communicated by S.E. Hambrusch

Abstract

Approximation algorithms for embedding hyperedges in a cycle so as to minimize the maximum congestion are presented. Our algorithms generate an embedding by transforming the problem into another problem solvable in polynomial time. One algorithm transforms it to a linear programming problem, and the other one to the problem of embedding edges in a cycle. Both algorithms generate an embedding with congestion at most twice of that in an optimal solution. Our problem has applications in CAD and parallel computation. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Approximation algorithms; Parallel computation; CAD; Linear time algorithms; Minimize congestion

1. Introduction

The problem of Embedding Hyperedges in a Cycle so a to Minimize Congestion (EHCMC) [3] has applications in design automation (routing nets around a rectangle [6–8,10], and moat routing [1,4,9,11], 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

$$E_C = \big\{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \text{ and } \{v_n, v_1\}\big\}.$$

A *connecting path or (simply a c-path)* P_i in C, for $1 \leqslant i \leqslant m$, is a path in C such that all the vertices in the ith hyperedge are in P_i. Our problem consists of finding c-paths P_i, for $1 \leqslant i \leqslant m$, such that the congestion which is measured by $\max_{e \in C} |\{i \text{ such that } e \in P_i\}|$ is least possible.

Frank et al. [2] developed a linear time algorithm for the EHCMC problem when the hypergraph is a graph (all hyperedges contain exactly two vertices). Ganley and Cohoon [3] showed that the EHCMC problem is NP-hard in general, but solvable 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., $\hat{f}_I \leqslant 3f_I^*$, 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

[1] Email: teo@cs.ucsb.edu.

0020-0190/98/$19.00 © 1998 Elsevier Science B.V. All rights reserved.
PII: S0020-0190(98)00117-3

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 (LP) transforms the problem to a linear programming problem, and the other one (LP-Free) transforms it to the problem of embedding edges in a cycle. The LP-Free algorithm takes linear time with respect to the total number of vertices in the hyperedges, but the LP algorithm is more robust because it may be easier to modify for minor variants of the EHCMC problem. Gonzalez [5] presents problem instances for which both of these algorithms generate solutions with congestion about twice of that in an optimal solution.

2. LP approximation algorithm

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., $h_{i,1} \leqslant h_{i,2} \leqslant \cdots \leqslant h_{i,m_i}$. We define the *adjacent paths (or simply a-paths)* $p_{i,j}$, for $1 \leqslant j \leqslant m_i$, 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

$$\sum_{j=1}^{m_i} a_{i,j} = m_i - 1, \quad 0 \leqslant i \leqslant m, \tag{1}$$

$$0 \leqslant a_{i,j} \leqslant 1 \quad \forall i \text{ and } j.$$

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 $x \geqslant 1/2$ 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

$$\sum_{p_{i,j} \text{ includes } e} a_{i,j} \leqslant k.$$

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.

Find Minimum k such that

$$\sum_{p_{i,j} \text{ includes } e} a_{i,j} \leqslant k \quad \forall e \in C,$$

$$\sum_{j=1}^{m_i} a_{i,j} = m_i - 1, \quad 0 \leqslant i \leqslant m, \tag{2}$$

$$0 \leqslant a_{i,j} \leqslant 1 \quad \forall i \text{ and } j.$$

Any feasible solution to any instance I of the EHCMC is a feasible solution to its corresponding LP program (2). Therefore, $f_I^* \geqslant k^*$, 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 may increase 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 $\hat{f}_I \leqslant 2f_I^*$. We shall refer to this procedure as algorithm LP.

Theorem 2.1. *The embedding generated for every instance I of the EHCMC problem by algorithm LP satisfies $\hat{f}_I \leqslant 2f_I^*$. The time complexity of this*

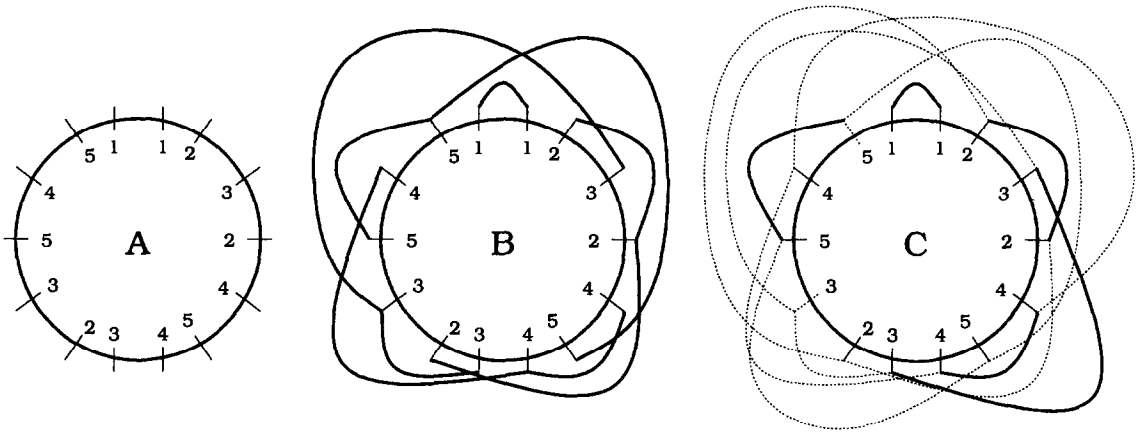

Fig. 1. The numbers inside the circle are the hyperedges that include those points.

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 all the numbers in the LP instance are integers between 0 and n, remember that n is the number of vertices in the hypergraph. □

An alternate LP formulation can be 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 $a_{i,j} = 1 - \bar{a}_{i,j}$. This LP formulation is simpler because the set of inequalities $0 \leqslant a_{i,j} \leqslant 1$ is reduced to $0 \leqslant \bar{a}_{i,j}$.

For brevity we only include the "seed" problem instance (Fig. 1(A)) which when generalized as in [5] defines a class of problem instances for which the LP algorithm generates solutions whose congestion is asymptotic to twice of that in an optimal solution. The "seed" problem instance has an optimal solution with congestion 3 (see Fig. 1(B)). An optimal solution to the LP problem is given in Fig. 1(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 bound is $\frac{5}{3}$.

[2] Actually this is the solution generated by using the simplex method to solve the LP problem.

3. LP-Free approximation algorithm

The main problem with the LP approximation algorithm is the time required to solve the linear programming 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.

Our algorithm, which we call LP-Free, transforms the problem instance 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 construct our solution to the original 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) in the obvious way. For example, the problem instance

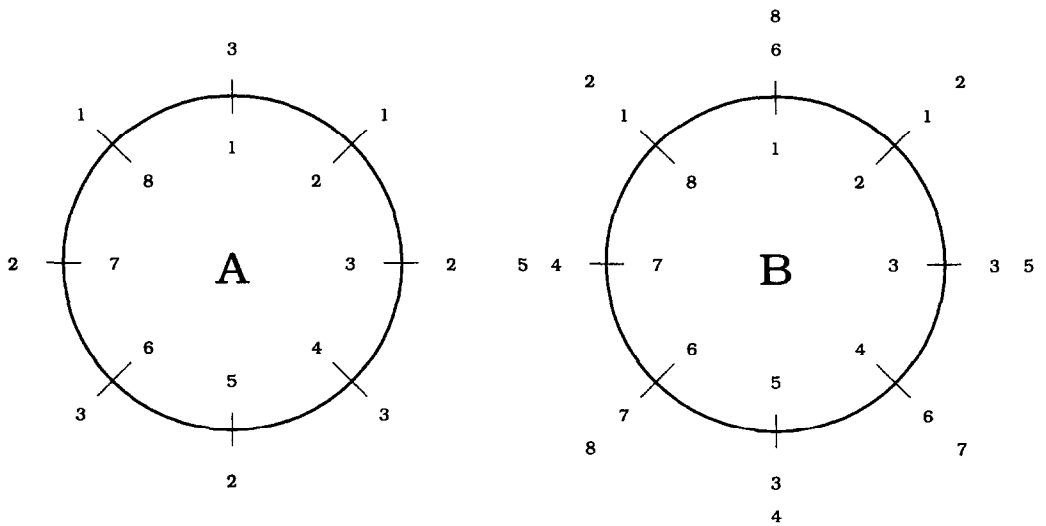

Fig. 2. The vertex numbers are inside the circle and the numbers outside indicate the hyperedges that including that vertex.

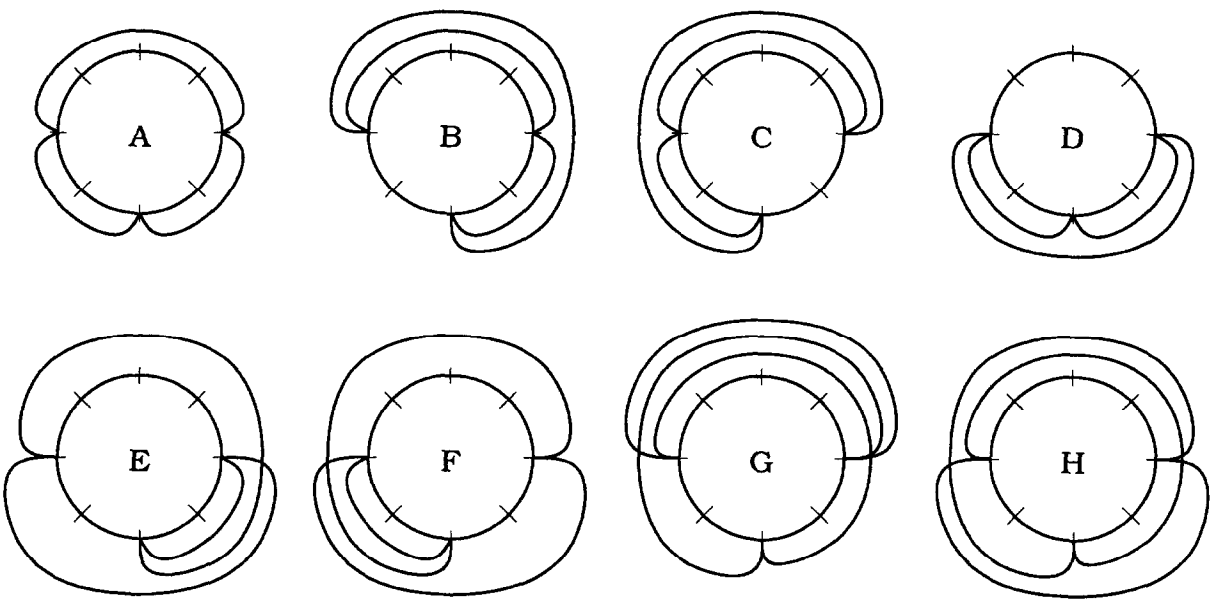

Fig. 3. Possible embeddings for the resulting edges for a hyperedge with three points.

in Fig. 2(A) is transformed to the instance given in Fig. 2(B). Note that hyperedge 1 in Fig. 2(A) becomes edges 1 and 2 in Fig. 2(C). This is represented by $1 \rightarrow 1, 2$. Similarly $2 \rightarrow 3, 4, 5$ and $3 \rightarrow 6, 7, 8$.

A hyperedge with three points is transformed into three edges and there are eight different ways all these edges could be embedded (see Fig. 3). However, embedding patterns E, F, G, and H in Fig. 3 can be converted to pattern A without increasing the congestion. Therefore, there exists an optimal solution where the hyperedge with three points is embedded as in Fig. 3 (A, B, C, or D). The same idea can be generalized to

all hyperedges. 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 the duplicate connections, or delete an a-path from a c-path that includes all its a-paths. The solution has congestion $\hat{f}_I \leqslant 2f_I^*$ simply because an optimal solution to the original problem can be transformed into a feasible solution to the resulting problem with congestion $2f_I^*$ and the algorithm in [2] gives the best possible solution to the resulting problem.

Theorem 3.1. *Algorithm LP-Free generates for every instance of the EHCMC a solution with congestion* $\hat{f}_I \leqslant 2f_I^*$. *The time complexity is linear with respect to the sum of the cardinalities of the hyperedges.*

Proof. By the above discussion. □

Both the LP and the LP-Free algorithms generate solutions with congestion about twice of that of an optimal solution for the class of problem instances given in [5].

4. Discussion

A slightly better approach results by applying the last transformation, but leaving out for each hyperedge one of the resulting 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 approximation bound, but, in general, a better solution may be obtained. In practice one could run several variations of our two algorithms, and then find and output the best of the solutions generated.

References

[1] B.S. Baker, R.Y. Pinter, An Algorithm for the optimal placement and routing of a circuit within a ring of PADS, IEEE FOCS (1983) 360–370.

[2] A. Frank, T. Nishizeki, N. Saito, H. Suzuki, E. Tardos, Algorithms for routing around a rectangle, Discrete Appl. Math. 40 (3) (1992) 363–378.

[3] J.L. Ganley, J.P. Cohoon, Minimum-congestion hypergraph embedding in a cycle, IEEE Trans. Comput. 46 (5) (1997) 600–602.

[4] J.L. Ganley, J.P. Cohoon, A provably good moat routing algorithm, in: Proc. Sixth Great Lakes Symposium on VLSI, 1996, pp. 86–91.

[5] T. Gonzalez, Improved approximation algorithms for embedding hypergraphs in a cycle, UCSB TR 97-23, 1997.

[6] T. Gonzalez, S.L. Lee, A linear time algorithm for optimal wiring around a rectangle, J. ACM 35 (4) (1988) 810–832.

[7] T. Gonzalez, S.L. Lee, A1.60 approximation algorithm for routing multiterminal nets around a rectangle, SIAM J. Comput. 16 (4) (1987) 669–704.

[8] A.S. LaPaugh, A polynomial time algorithm for optimal routing around a rectangle, in: Proc. 21st IEEE FOCS, 1980, pp. 282–293.

[9] R.K. McGehee, A practical moat router, in: Proc. 24th Design Automation Conference, 1987, pp. 216–222.

[10] M. Sarrafzadeh, F.P. Preparata, A bottom-up layout technique based on two-rectangle routing, Integration: VLSI J. 5 (1987) 231–246.

[11] C.C. Tsai, S.J. Chen, A linear time algorithm for planar moat routing, J. Inform. Sci. Eng. 10 (1) (1994) 111–128.