A Provably Good Moat Routing Algorithm

Joseph L. Ganley*†
Cadence Design Systems, Inc.
555 River Oaks Parkway
San Jose, California 95134

James P. Cohoon*
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

Abstract

Moat routing is the routing of nets between the input/output pads and the core circuit. In this paper, it is proved that moat routing is NP-complete under the routing model in which there are no vertical conflicts and doglegs are disallowed (i.e., every net is routed within a single track). This contrasts with the fact that channel routing is efficiently solvable under these restrictions. The paper then presents an approximation algorithm for moat routing that computes moat routing solutions that are guaranteed to use at most four times the optimal number of tracks. Empirical results are presented indicating that for a number of industrial benchmarks, the algorithm produces solutions that are near optimal and that use significantly fewer tracks than previous moat routing strategies.

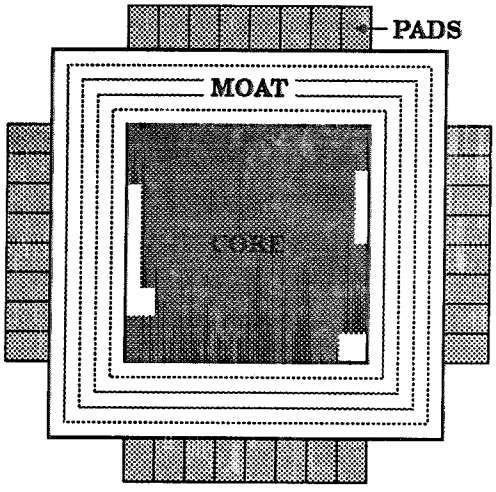


Figure 1: A moat routing instance.

1 Introduction

The final stage in detailed routing is typically to route the connections between the input/output pads and the core circuit. The area between the core and the pads is called the *moat*, and this routing task is consequently called *moat routing*.

A moat routing instance consists of a number of nets whose pins lie on either or both of the inside perimeter of the padframe and the outside perimeter of the core circuit area. The moat between the pads and the core is divided into a number of concentric *tracks*, similar to a channel routing instance except that each track forms a circle rather than a line segment. A set of pads, a circuit core, and the moat between them are illustrated in Figure 1.

In this paper we assume the use of a routing model in which doglegs are not allowed, i.e., every net is routed within a single track. Furthermore, we assume that there are no vertical constraints (as Wang [11] points out, the pins on the pads are typically spaced sufficiently far apart that any vertical constraints can be eliminated). We henceforth refer to this model simply as the *restricted routing model.*

As in previous works [8, 11], we use a 2-layer model in which the tracks lie in one layer and the radial connections between the core or pad pins and the tracks lie in the other layer.

The remainder of this paper is organized as follows. Section 2 describes some concepts regarding intersection graphs and channel routing algorithms. In Section 3, it is proved that under the restricted routing model, moat routing is NP-complete (whereas the restricted routing model renders channel routing efficiently solvable). Section 4 describes a technique for determining a lower bound on the number of tracks required for a given moat routing instance. Section 5 then describes an approximation algorithm that computes a moat routing that uses at most four times the optimal number of tracks. In Section 6, empirical evidence is provided indicating that for a number

*Partially supported by National Science Foundation grants MIP-9107717 and CCR-9224789.

†Partially supported by a Virginia Space Grant Fellowship.

0-8186-7502-0/96 $5.00 © 1996 IEEE

of industrial benchmarks, the approximation algorithm performs well with respect to lower bounds and previous moat routing strategies. Finally, Section 7 concludes with some ongoing work.

2 Terminology

The *graph K-colorability* problem is defined as follows: given a graph, is it possible to assign a color to each vertex such that at most K colors are used and such that the endpoints of every edge are colored differently?

An *interval graph* is a graph in which the vertices correspond to intervals on a line and in which there is an edge between every pair of vertices whose intervals intersect[1]. Under the restricted routing model, the channel routing problem corresponds directly to the problem of coloring an interval graph. Each interval corresponds to a net, and its endpoints are the minimum and maximum x coordinates of the pins in the net (assuming without loss of generality that the channel is horizontal). A K-coloring of this interval graph corresponds directly to a channel routing solution using K tracks. Each color corresponds to a track, and since no pair of intervals of the same color intersect, all intervals of like color can be routed within a single track. The K-coloring problem can be efficiently solved in an interval graph, and thus a channel routing solution that is optimal within the restricted routing model can be efficiently computed.

One classic algorithm that does so is the *left-edge algorithm* of Hashimoto and Stevens [7]. The left-edge algorithm proceeds as follows: sort the intervals according to the x coordinates of their left endpoints. Then process the nets in this sorted order, inserting intervals into tracks in a greedy fashion: each interval is inserted into the first track in which it fits, or if it fits in none of the current tracks, then it is inserted into a new track.

The *density* of a channel routing instance is the maximum number of intervals that intersect any vertical line, i.e., the size of a maximum clique in the corresponding interval graph. Clearly the density of an instance is a lower bound on the number of tracks required to route the instance. In fact, interval graphs are *perfect graphs*, meaning that the maximum clique size is equal to the minimum number of colors required to color the graph. Thus, the optimal number of tracks is precisely equal to the density, and the left-edge algorithm computes an optimal channel routing solution.

A *circular arc graph* is similar to an interval graph except that the vertices correspond to arcs on a circle rather than intervals on a line. Unlike interval graphs, circular arc graphs are not perfect, meaning that the minimum number of colors required to color a circular arc graph is not necessarily equal to its density (though density is still clearly a lower bound). One might suspect that moat routing is analogous to coloring a circular arc graph in the same manner that channel routing is analogous to coloring an interval graph. In the next section, we prove that a moat routing algorithm can indeed be used to solve the K-colorability problem in a circular arc graph. Unfortunately, since circular arc graph coloring is NP-complete [4], this reduction implies that moat routing under the restricted routing model is NP-complete as well. This contrasts with the fact that channel routing is efficiently solvable.

3 Computational Complexity

In this section, we prove that a moat routing algorithm can be used to solve the K-coloring problem in a circular arc graph. Since coloring a circular arc graph is NP-complete [4], moat routing is NP-complete as well.

Theorem 1 *Moat routing is NP-complete under the restricted routing model.*

Proof: Inclusion in NP is obvious. NP-completeness is shown by a reduction from circular arc graph coloring.

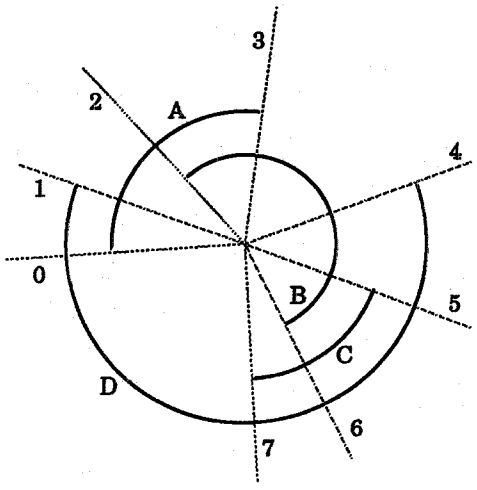
An instance of the coloring problem for circular arc graphs is a circular arc graph G and an integer K. The question is whether G can be colored using K or fewer colors.

Assume without loss of generality that the endpoints of the arcs that comprise G are all distinct [6]. Sort the endpoints of the n arcs in clockwise order, and call them $p_1, p_2, \ldots, p_{2n}$. We now build a moat routing instance that mirrors the structure of G.

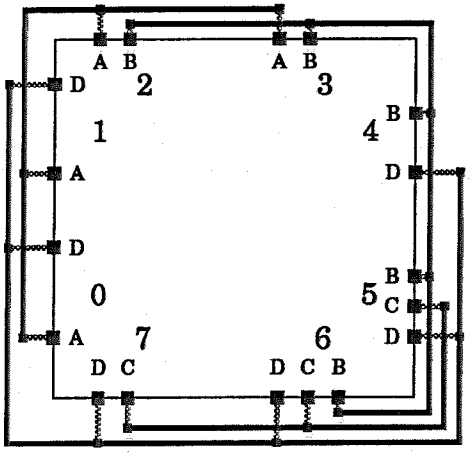
All pins in the moat routing instance lie on the perimeter of the core. They are arranged in $2n$ groups, each of which contains at most n pins. The groups are arranged around the core perimeter in the same order as the corresponding points on the circle. Let t_{ij} denote the j^{th} pin in the i^{th} group of pins. A net R_j is built for each arc $a_j = (p_L, p_R)$ in G, in which the pins are t_{ij} for all $p_L \leq i \leq p_R$ (index arithmetic is

[1] All graph-theoretical concepts discussed here are described in Golumbic [5].

performed modulo $2n$, i.e., $a < b$ is understood to imply that a is counterclockwise of b). This construction is illustrated in Figure 2.



(a)



(b)

Figure 2: (a) A 3-colorable circular arc graph and (b) a corresponding 3-track moat routing instance.

We claim that a K-track moat routing solution to the construction described above exists if and only if G is colorable using K colors.

($\Rightarrow$) *If G is colorable using K colors, then a K-track solution exists for the moat routing instance constructed as described above.* For each set of arcs given the same color, route the corresponding nets within a single track. Such a routing is valid since no two arcs with the same color intersect.

($\Leftarrow$) *If a K-track solution exists for the moat routing instance constructed as described above, then G is colorable using K colors.* The routing for each net is an entire track minus the section between two adjacent pins. If the missing section lies between pins t_{ij} and $t_{i+1,j}$ such that $p_L \leq t_{ij} < t_{i+1,j} \leq p_R$, then the net must intersect the routing of every other net, and thus it is routed within its own distinct track. Thus, any net whose routing does not correspond to the circular arc from which it was constructed can be rerouted so that it does correspond to its circular arc without increasing the number of colors required. Therefore, we can transform any K-track moat routing solution into a K-track solution in which the routing of each net corresponds directly with the circular arc from which it was constructed. Each arc is then given a color corresponding to the track in which its corresponding net is routed, and the resulting coloring is a valid K-coloring of G.

This transformation is performed in $O(n^2)$ time. Since the circular arc graph coloring problem is NP-complete [4] and is reducible in polynomial time to the moat routing problem, the moat routing problem is NP-complete. □

4 Lower bounds

Due to the circular nature of the moat, many ideas from the channel routing literature cannot be applied directly. In a (horizontal) channel, a routing algorithm must assign a track to each net, but the horizontal span of the routing is determined by the instance, i.e. there is only one possible horizontal span for each net. For moat routing, the circular moat creates a possibility of many different routing paths, independent of the assignment of nets to tracks. Specifically, a net with m pins can be routed in m different ways, each corresponding to a complete track with the span between two adjacent pins removed.

Since nets in a moat routing instance can be routed in multiple ways, the notion of density as it applies to channel routing cannot be directly generalized to moat routing. However, we derive a slightly different lower bound by considering pairs of points around the moat. Consider a pair of lines, each of which extends from the core circuit area perpendicular to its border. Each such pair separates the moat into two channels.

Find a pair of such lines such that the number of nets with pins in both channels is maximum. We say that these nets are *cut* by the pair of lines, and denote the maximum number of cut nets as N. Each cut net must use a track that intersects one of the lines, so

the minimum number of tracks required for the moat routing is at least $\lceil N/2 \rceil$.

5 An Approximation Algorithm

Further exploration of the ideas in the previous section leads to an approximation algorithm for the moat routing problem. As in Section 4, find a pair of lines that cuts a maximum number of nets. These lines divide the moat into a pair of channels; call them the *left* and *right* channels. We now prove that if N nets are cut by the maximum cut, then the remaining uncut nets can be routed using at most N tracks.

Lemma 1 *If N nets are cut by the maximum cut, then the uncut nets can be routed using at most N tracks.*

Proof: Suppose to the contrary that N nets are cut by a maximum cut but the uncut nets cannot be routed using N tracks. Since the nets in the left channel and the nets in the right channel do not intersect, it must be the case that either the right or left channel cannot be routed using N tracks. Suppose without loss of generality that the left channel cannot be routed using N tracks. As mentioned in Section 2, if the density of a channel (the maximum number of nets that span any point in the channel) is D, then the channel can be routed using D tracks. Therefore, by our assumptions, the density of the left channel exceeds N. However, the line in the left channel that cuts more than N nets and another line anywhere in the right channel form a cut that cuts more than N nets, contradicting the fact that N nets are cut by the maximum cut. □

Each of the two sets of uncut nets forms a channel routing instance, so by Lemma 1, the uncut nets can be routed in at most N tracks using, for example, the left-edge algorithm described in Section 2. Suppose the cut nets are routed arbitrarily; since there are N of them, they can be routed in at most N tracks. Thus, the moat routing computed by the algorithm uses at most $2N$ tracks. As discussed in Section 4, the minimum number of tracks required to route an instance is $\lceil N/2 \rceil$. Thus, the algorithm computes a solution that uses at most $2N/\lceil N/2 \rceil \leq 4$ times the optimal number of tracks[2].

[2] As an aside, a similar approach results in a 2-approximation algorithm for coloring a circular arc graph G. Find a maximum clique C in G, and color it using $|C|$ colors. Color the interval graph $G - C$ optimally, which requires at most $|C|$ colors. Thus, a coloring using at most $2|C|$ colors is computed, and the optimal coloring requires at least $|C|$ colors.

Suppose there are n nets containing a total of m pins. A maximum cut is found in $O(m^2)$ time. The nets are then routed in at most $O(n \log n)$ time. Thus, the total time complexity is $O(m^2 + n \log n) = O(m^2)$.

Though an arbitrary routing of the cut nets is sufficient to satisfy the approximation bound, in practice one would like to compute a good routing. We turn once again to circular arc graphs to devise an effective heuristic for routing the cut nets.

Construct a circular arc graph G containing at most nm arcs as follows: For each net R_i, denote the pins in R_i as $t_0, t_1, \ldots, t_{|R_i|-1}$, in clockwise order. For each R_i, add to G the arcs $[t_{(j+1) \bmod |R_i|}, t_j]$ for all $0 \leq j < |R_i|$. Note that each arc includes all the pins in R_i. Note also that if $|R_i| > 2$, then the arcs constructed from R_i are all pairwise intersecting.

Now find a *maximum independent set (MIS)* in G. This is accomplished in $O((nm)^2)$ time, since there are at most nm arcs in G [6]. Since all intervals in a net with 3 or more vertices are pairwise intersecting, the MIS cannot contain more than one arc from a net R_i unless $|R_i| = 2$. If it contains both arcs from a net R_i with $|R_i| = 2$, then the size of the MIS is 2, and an MIS that does not contain two arcs from the same net can be found exhaustively in $O(n^2)$ time. Thus, an MIS that does not contain 2 arcs from the same net is computed in $O((nm)^2)$ time.

Once the MIS has been computed, route each net for which there is an arc in the MIS according to that arc. Remove all arcs that were constructed from nets thus routed, and repeat the process until no arcs remain. We call this heuristic the *iterated maximum independent set (IMIS)* heuristic. Since each pass requires $O((nm)^2)$ time, and at most $O(n)$ passes are performed, the IMIS heuristic runs in $O(n(nm)^2)$ time.

In order to better use the space among the uncut nets, we have implemented the algorithm as follows: G contains at least one arc for every net. For a cut net, G contains the arcs corresponding to every possible routing path, as described above. For an uncut net, G contains only the single arc corresponding to its routing within the left or right channel.

Using the IMIS heuristic to route the nets in the approximation algorithm increases the time complexity of the approximation algorithm to $O(m^2 + n(nm)^2) = O(n(nm)^2)$.

6 Experimental Results

We have implemented our approximation algorithm in order to compare its performance in practice with