Planar Topological Routing [*]

Andrew Lim[†] Venkat Thanvantri[‡] Sartaj Sahni[§]

Abstract

We develop a simple linear time algorithm to determine if a collection of two-pin nets can be routed, topologically, in a plane (i.e. single layer). Experiments indicate that this algorithm is faster than the linear time algorithm of Marek-Sadowska and Tarng [1]. Topological routability testing of a collection of multi-pin nets is shown to be equivalent to planarity testing and a simple linear time algorithm is developed for the case when the collection of modules remains connected following the deletion of all nets with more than two pins.

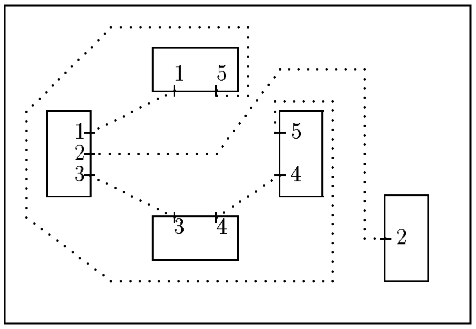
Keywords and Phrases
single layer topological routing

[*]This research was supported in part by the National Science Foundation under the grant MIP 91-03379

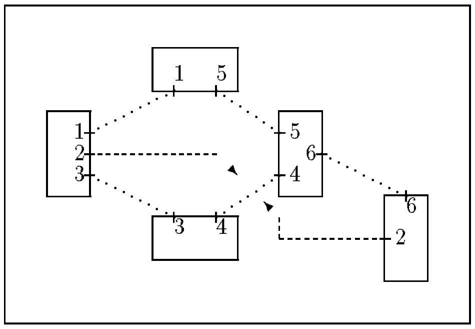
[†]Information Technology Institute, National Computer Board, 71 Science Park Drive, Singapore 0511, Republic of Singapore.

[‡]Department of Computer and Information Sciences, University of Florida, Gainesville, FL 32611, U.S.A.

[§]Department of Computer and Information Sciences, University of Florida, Gainesville, FL 32611, U.S.A.



(a) Planar Routable



(b) A non planar routable example

Figure 1: A planar routable and a non planar routable case

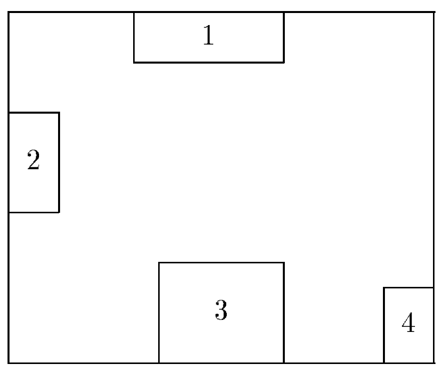
1 Introduction

The problem of routing two-pin nets on a single layer has been studied previous by several researchers. The river routing and switch box routing problems are special cases of this. Efficient algorithms for these can be found in [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. In this paper, we are concerned with the problem of routing (topologically) a collection of two-pin nets in a single layer or plane. We refer to this problem as the **TPR** problem. The input to the problem is a two dimensional routing surface with a collection of modules placed in it (Figure 1(a)). We assume that no two modules touch. There are pins on the periphery of the modules. Pins with the same number define a net and are to be joined by an interconnect or wire. In topological routing, we are concerned with defining wire paths. However, no underlying grid is assumed and there is no minimum wire separation requirement. Thus wire paths can take any planar shape and may run arbitrarily close to each other. Wires are not permitted to intersect or run over modules. In Figure 1(a), the broken lines indicate wire paths. The routing instance (RI) of Figure 1(a) is topologically routable in a single layer while that of Figure 1(b) is not. The TPR problem for RIs in which all modules lie on the boundary of the routing region (or more precisely all pins are on the boundary of the region) was studied in [1, 2, 8]. A simple linear time algorithm for this version of the TPR problem was developed in these papers For the case in which none of the modules are on the boundary, Pinter [8] has suggested using the linear time planarity testing algorithm of Hopcroft and Tarjan [12]. His algorithm is quite complex. Marek-Sadowska and Tarng [1] have considered the TPR problem and several variants which include flippable modules and multi-terminal nets. They develop a linear time algorithm for TPR which is based on module merging. In this paper, we develop, in section 3, another linear time algorithm for the general TPR problem that is almost as simple as the one of [1, 2, 8] for the restricted TPR problem. Experimental results presented in section 5 indicate that our algorithm is considerably faster than the TPR algorithm of [1] particularly if the routing instance is not planar routable. For the case of multi-pin nets, we show, in section 4, that testing for topological routability is equivalent to graph planarity testing and that finding the maximum number of nets that is topologically routable is NP-Complete. We also extend our two-pin algorithm to handle multi-pin instances in which the modules remain connected following the deletion of all nets with more than two pins.

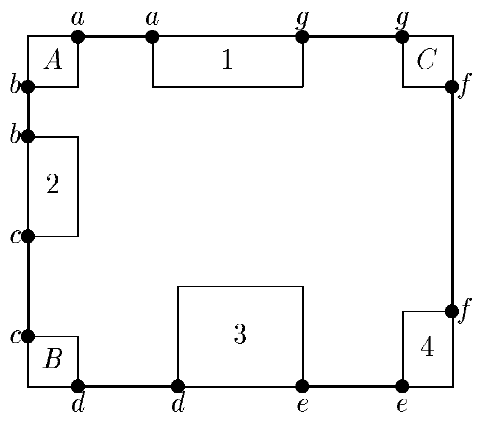
2 Preliminaries

To simplify matters, we shall assume that TPR RIs that have modules on the boundary (Figure 2(a)) have been augmented by a set of nets that are required to be routed on the boundary and that this routing together with the module boundaries enclose the routing region (Figure 2(b)). This augmentation may require the addition of corner modules (A, B, C of Figure 2(b)). This assumption is needed so that our algorithm can account for the constraint that one cannot route around a boundary module but can route around all other modules.

A *pin segment*, $P = p_1p_2 \ldots p_k$, is a sequence of pins on the boundary of a module. $p_1 \ldots p_k$ appear in this order when the module is traversed counter-clockwise beginning at p_1. Some of the pin segments of the modules of Figure 3 are: $abcde$ and $gjkH$ of module 1; MLK and $LKJGf$ of module 3; and AiF of module 2. Let $last(P)$ and $first(P)$, respectively, denote the last and first pins of segment P. Let $net(p_i)$ denote the net associated with pin p_i. Note that two pins p_i and p_j are to be connected by a wire iff $net(p_i) = net(p_j)$. A curve, $C = P_1P_2 \ldots P_j$, is a sequence of pin segments such that $net(last(P_i)) = net(first(P_{i+1}))$, $1 \leq i < j$. A curve, $C = P_1P_2 \ldots P_j$, is a *closed curve* iff $net(last(P_j)) = net(first(P_1))$. In Figure 3, $net(p_i)$ is the lowercase letter corresponding to p_i. So, $net(h) = net(H) = h$. Some of the curves of Figure 3 are $Ih\ Habcdeg\ Gf\ FEDCBAi$, $j\ JGfM\ mIh$, $edcba\ ABCDE$ and $ABC\ cdeg\ GfM$. $IhHabcdeg\ Gf\ FEDCBAi$ and $edcba\ ABCDE$ are closed curves. With any curve $C = P_1P_2 \ldots P_j$, we associate $j - 1$ (j in case C is closed) wires. These, respectively, connect the pins $last(P_i)$ and $first(P_{i+1})$, $1 \leq i < j$ (and $last(P_j)$ and $first(P_1)$ in case of a closed curve). Note that the curves, closed curves, and wires associated with any



(a) RI



(b) Augmented RI

Figure 2: Augmentation

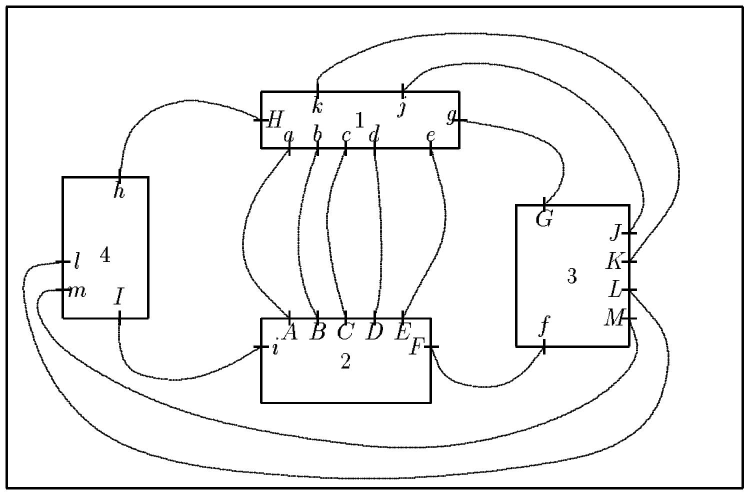
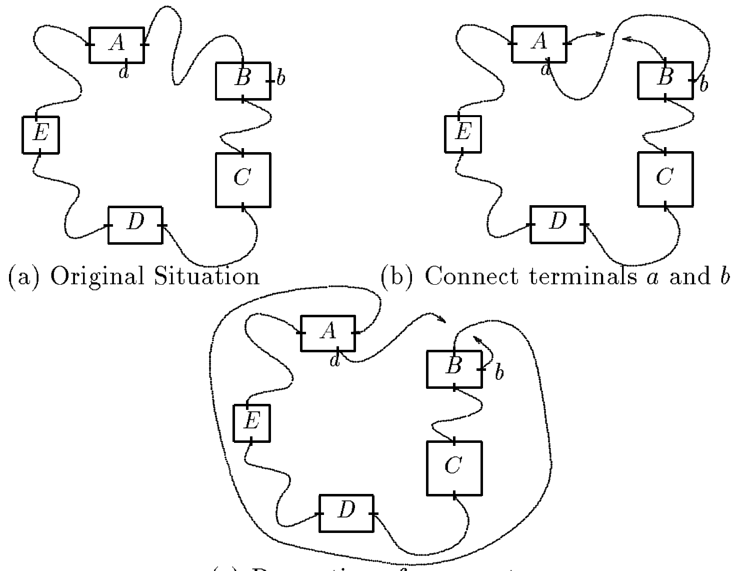


Figure 3: An example to illustrate some terminology



(a) Original Situation (b) Connect terminals a and b

(c) Re-routing of some net

Figure 4: Two possibilities to connect a and b

RI depend only on the modules and the net to pin assignments. These are not a function of the layout of any of the wires.

For any closed curve $C = P_1 P_2 \ldots P_j$ we define the following:

$module(P_i)$	...	module corresponding to pin segment P_i
$pins(module(P_i))$	...	set of all pins on module $module(P_i)$
$pins(P_i)$	...	set of all pins on segment P_i
$pins(C)$	...	set of all pins on curve $C = \bigcup_{i=1}^{j} pins(P_i)$
$ext\_pins(C)$	...	$\bigcup_{i=1}^{j} pins(module(P_i)) - pins(C)$

Note, it is possible that $module(P_i) = module(P_j)$, for $i \neq j$.

Lemma 1 : *Let I be an RI that contains a closed curve C with respect to which there are two pins $a \in pins(C)$ and $b \in ext\_pins(C)$ such that $net(a) = net(b)$. I is not planar routable.*

Proof : Figure 4 shows two possibilities. It should be clear that no matter how the wires

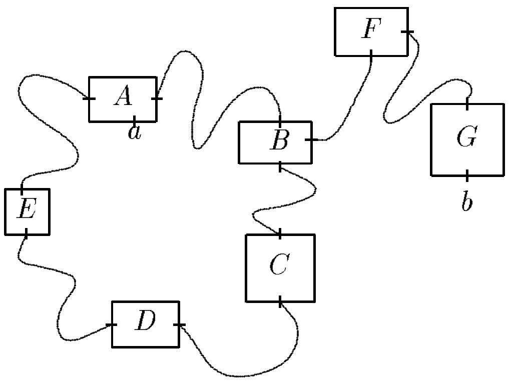


Figure 5: Another not planar routable situation

of C and the wire (a, b) are laid out, there must be an intersection between two of these. $\square$

Lemma 2 : *Let I be an RI that contains a closed curve $C = P_1, P_2, \ldots, P_j$ and another curve $R = R_1 R_2 \ldots R_k$ such that $module(R_1) = module(P_d)$ for some d, $1 \leq d \leq j$ and $first(R_1) \in ext\_pins(C)$ (see Figure 5). Assume that there exist two pins a and b such that $a \in pins(C)$, $b \in ext\_pins(C) \bigcup pins(R)$, and $net(a) = net(b)$. I is not planar routable.*

Proof : Follows from Lemma 1. $\square$

Two modules are *connected* iff there is a curve $C = P_1 P_2 \ldots P_j$ such that both modules are in $\bigcup_{i=1}^{j} module(P_i)$. A *connected component* (or simply component) is a maximal set of modules that are pairwise connected. It is easy to see that the connected components of an RI are disjoint. A *boundary component* is a connected component that includes at least one boundary module. Note that an RI with no boundary modules has no boundary component while an RI with at least one boundary module has exactly one boundary component (this is because RIs with boundary components have been augmented as in Figure 2(b)).

Lemma 3 : *An RI is topologically routable iff its components are (independently) topologically routable.*

Proof : It is easy to see that if the RI is topologically routable then each of its components is topologically routable. Assume that each component is topologically routable. Order the

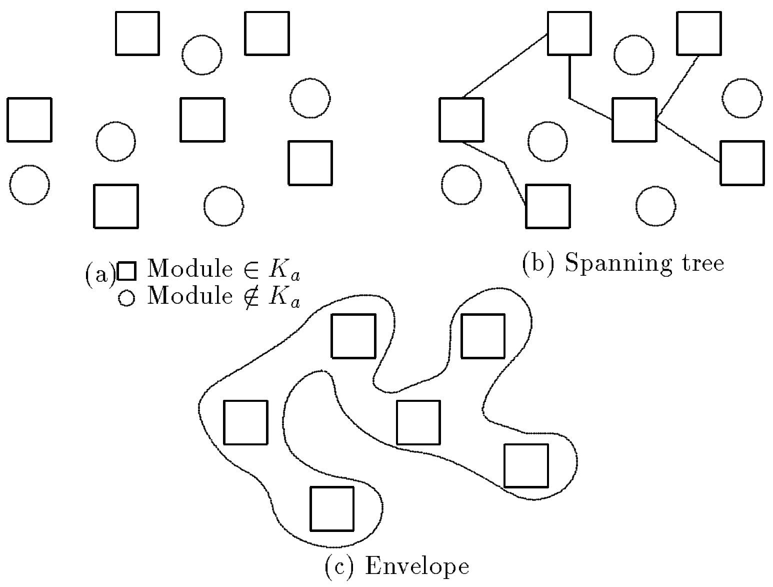
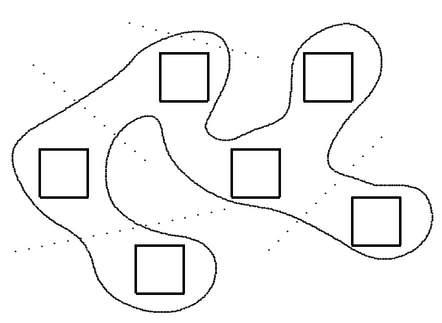


Figure 6: Constructing the envelope of a component

components of the RI so that the boundary component is first. The remaining components are in arbitrary order. Let the components in this order be $K_1, K_2, \ldots, K_k$. If $k = 1$, then nothing is to be proved. So, assume $K > 1$. We shall show how to construct a topological routing for $K_1, K_2 \ldots, K_a$ from a topological routing for $K_1, \ldots, K_{a-1}$ and K_a, $2 \leq a \leq k$. First since $a > 1$, K_a is not a boundary component. So, it is possible to surround it by a closed non self intersecting line such that the region enclosed by this line includes exactly those modules that are in K_a and no module touches the line. The region enclosed by this closed line has the property that any two points in the enclosed region can be joined by a line (not necessarily straight) that lies wholly within the region. We refer to the surrounding line as the *envelope* of K_a. One way to obtain an envelope of K_a is to first construct a set of $|K_a| - 1$ ($|K_a|$ is the number of modules in K_a) lines (not necessarily straight) so that modules of K_a together with these lines form a connected component in the graph theoretic sense (see Figure 6). These lines do not touch or cross any of the modules of RI. This construction can be done as every pair of modules of an RI can be can be connected by such a line. The lines and modules define a spanning tree for K_a. By fattening the lines as in Figure 6(c), the envelope is obtained. It is easy to see that


(a) Intersections

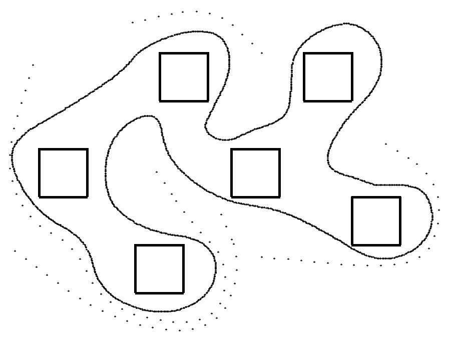

(b) Re-routing

Figure 7: Re-routing to free independent component

if K_a is topologically routable, then it is topologically routable with the defined envelope. So, use such a topological routing for K_a. When this routing is embedded into the routing for $K_1, \ldots, K_{a-1}$ some of the topologically routed wires of $K_1, \ldots, K_{a-1}$ may intersect (or touch) the envelope of K_a. However, none of these wires originate or terminate in the envelope of K_a. So, these can be rerouted following the contour of the envelope (Figure 7). $\square$

As a result of Lemma 3, we need concern ourselves only with the case when the RI has a single component.

3 The Algorithm

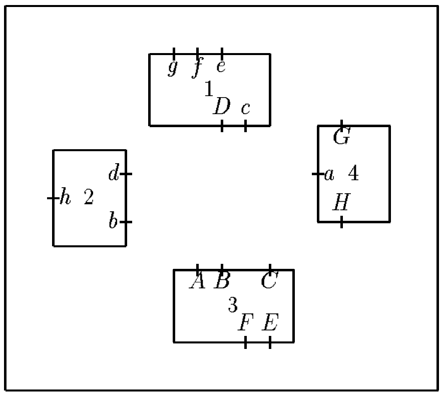
Our algorithm to obtain a topological routing of a component uses Lemmas 1 and 2 to detect infeasibility. The algorithm is given in Figure 8. As stated, it only produces an ordering of the wires such that when the wires are topologically routed, one at a time, in this order, then there is always a path between the two end points of the wire currently being routed such that this path does not intersect previously routed wires or cross any of the modules. This is sufficient to obtain the actual topological routing.

Our algorithm employs two stacks A and B. Stack A maintains a pin sequence that defines a curve of the RI. Stack B is used to retain pins that define closed curves with

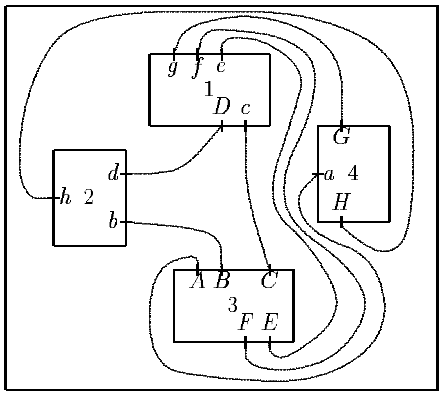
Algorithm *Testing_Planar_Routability*

```
Step 1: Let $m$ be any module of the component and let $p$ be any pin of $m$.

Step 2: Examine the pins of $m$ in counterclockwise order beginning at pin $p$. When a pin
    $q$ is being examined compare $net(q)$ and $net(r)$ where $r$ is the pin (if any) at the top
    of stack $A$. If stack $A$ is empty or $net(q) \neq net(r)$ then add $q$ and the remaining pins
    of $m$ to the top of stack $A$. Otherwise output $(q, r)$ and unstack $r$ from $A$.

Step 3: If both stack $A$ and $B$ are empty, then terminate.

Step 4: Let $r$ be the pin at the top of stack $A$. Let $s$ be the pin such that $net(r) = net(s)$.

    (a) If $s$ is at the top of the stack $B$, then [output $(r, s)$; unstack $r$ from $A$ and $s$
        from $B$; go to start of Step 3].

    (b) If $s$ is in stack $B$ but not at the top, then [output("The RI is not planar
        routable"). Terminate].

    (c) If $s$ is in stack $A$, then [unstack $r$ from A; add $r$ to stack $B$; go to the start of
        Step 4].

    (d) If $s$ is in neither of the stacks then [ set $p$ to $s$; let $m$ be the module containing
        $s$; go to Step 2].
```

Figure 8: Topological routing.



(a)Example RI



(b)A possible topological routing

Figure 9: Example RI

respect to a (sub) curve on stack A. We describe the working of the algorithm with the aid of an example (Figure 9(a)). There are four modules $1-4$ and 16 pins $a-h$ and $A-H$. $net(p) = p$ if p is a lowercase letter and $net(p) = lowercase(p)$ if p is an uppercase letter. Suppose we begin in step 1 with $m = 3$ and $p = B$. Then in step 2, $BAFEC$ get stacked, in that order on to stack A. This corresponds to the curve of Figure 10(a). Pin c is in neither of the stacks and in step 4(d), we set $m = 1$, $p = c$, and go to step 2. In this step, the wires Cc, Ee and Ff are output for routing. The pins g and D are put on the top of stack A. The curve traced so far is shown in Figure 10(b). The routed wires are also shown as a curve. Note that these wires have to be routed using the procedure $find\_route$, otherwise they can enclose a non-empty region. The curve is extended to module 2 and stack A has configuration $BAghb$. The wire Dd is output for routing. The curve has the form as shown is Figure 10(c). The curve cannot be extended further as both end points of wire Bb are on the stack. This means that we have detected a closed curve of the RI. The detected curve is that of Figure 10(c). We defer the routing of Bb until we have verified emma 1 and 2 for this closed curve. The deferment also ensures that the current topological routing does not contain a closed line. If Bb were routed now, then the wires Bb, Cc, and Dd together with the boundaries of modules 1, 2, and 3 would define a closed line that encloses a non-empty region. This could result in future routing problems as there would be no path between a point in the region and one that is outside the region. For example, if the routing of

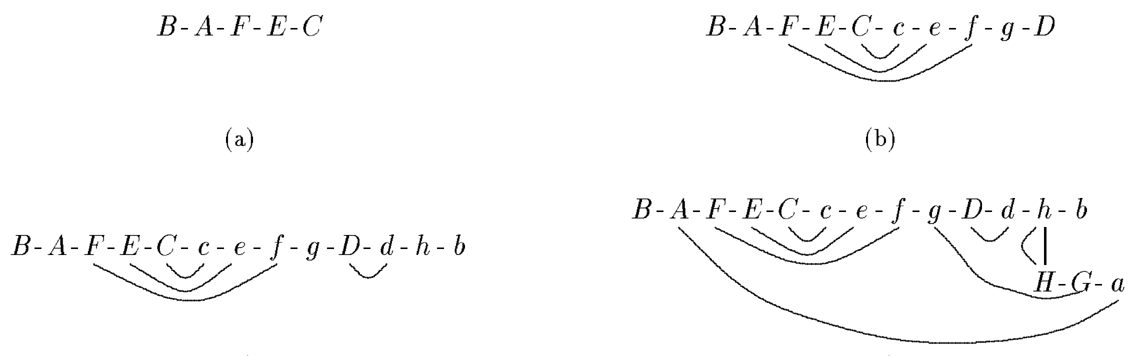


Figure 10: Illustration of the routing sequence

Figure 11 is used, then there is no path between a and A as a is in the enclosed (shaded) region while A is outside of it. The routing of Bb is deferred by saving b on stack B. The curve of stack A is extended to module 4 via the wire hH. Wire hH is output for routing. Also the wires gG and Aa are output for routing. Stack A contains the pin B and stack B contains the pin b. The curve is shown in Figure 10(d). Finally, the wire Bb is output for routing and the since both the stacks are empty, the algorithm terminates successfully in step 3.

The routing order is Cc, Ee, Ff, Dd, hH, gG, Aa and Bb. Let us try this out on our example. We see that no matter how Cc is routed there will remain a routing path for the remaining wires. The routing of Dd and Hh cannot create any enclosed regions and so cannot affect the feasibility of future routes. When Ee and Ff are routed, an enclosed region can be formed. Hence these wires have to be routed using the procedure *find_route* of Figure 13, otherwise they can enclose a non-empty region. The topological routed RI can be found in figure Figure 9(b).

Lemma 4 : *If algorithm Testing_Planar_Routability terminates in step 3, then the input instance is topologically routable.*

Proof : We shall show that the algorithm Testing_Planar_Routability maintains the following invariant:

There is a topological routing of all wires output so far such that each remaining wire is

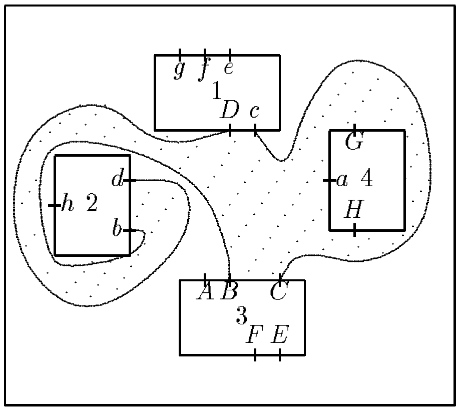


Figure 11: Trapped terminal and module

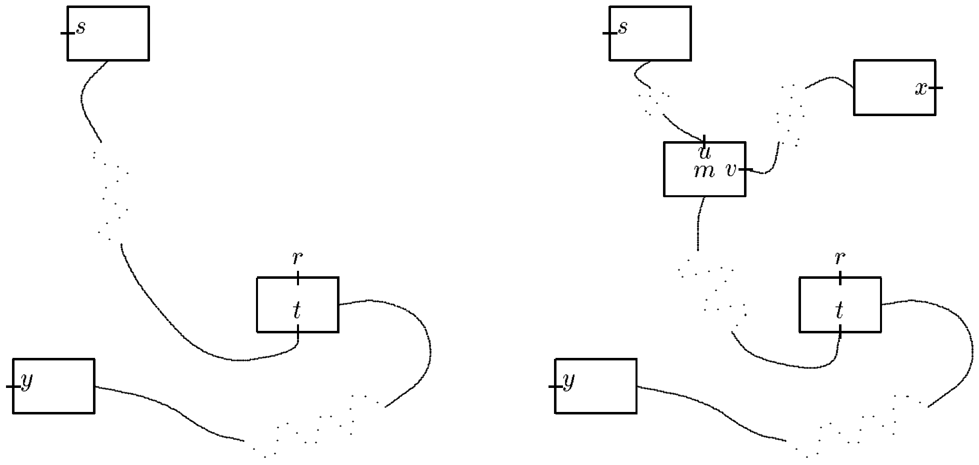


Figure 12: To illustrate conflict

(individually) topologically routable.

This is true when we start as at this time, no wires have been output and for each wire, there is a routing path between its two pins. Assume that the invariant holds just before some wire (r, s) is output. We shall show the invariant holds after this wire is output for routing. Wire (r, s) satisfies exactly one of the following:

(a) It is output in step 2, r is a pin that was on stack A, s is the first pin to be reached on its module.

(b) It is output in step 2, r was on stack A, s is not the first pin to be reached on its module.

(c) It is output in step 4(a). At the time of output, r was at the top of stack A and s at the top of stack B.

If we are in case (a), then since $module(s)$ is reached for the first time, no matter how wire (r, s) is routed at this time no new enclosed regions are formed. Hence all remaining wires remain routable.

The proofs for cases (b) and (c) are similar. We consider case (c) only. From the algorithm it follows that at some time prior to the output of (r, s), both r and s were on stack A, s was at the top of A and about to be moved to stack B. The pins on stack A beginning with r and ending at s define a closed curve C (as $net(r) = net(s)$). Let these pins be from modules $module(r) = M_1, M_2, \ldots, M_k = module(s)$ (in this order moving up stack A). Let p be any pin in $pins(C) - \{r, s\}$ and let q be such that $net(p) = net(q)$. We may assume that either $q \in pins(C)$ or $module(q)$ is unvisited at the time s is moved from stack A to stack B. Note that if this is not the case, then q is in stack A and below r at this time. From the working of the algorithm it follows that when r reaches the top of stack A (as it does by the assumption of (r, s) being output from step 4(a)), p must be on stack B and above A. So, the algorithm should have terminated unsuccessfully in step 4(b), contradicting the assumption of termination in step 3.

Let U be the set of unvisited modules at the time s is transferred from stack A to stack B. By extending our previous argument, we see that the set N of modules visited by the algorithm between the time s is transferred from stack A to stack B and the time (r, s) is output is such that

(1) $N \subseteq U \cup modules(C)$.

```
Algorithm find_route(r, s)
begin
    $currentpin := s$; $l :=$ clockwisepin($currentpin$);
    while ($l \neq r$) do
    begin
        Step 1: Route clockwise from $currentpin$ to $l$ following the module boundary
        Step 2: $currentpin = q$ such that $net(q) = net(l)$
        Step 3: Continue the route from $l$ to $currentpin$ following the existing route
            closely
        Step 4: $l =$ clockwisepin($currentpin$)
    end
    Complete the route from $currentpin$ to $l = r$ following the module boundary
end
```

Figure 13: Algorithm to find routing path between pins r and s

(2) All pins in $(N \cap U) \cup (pins(C) - \{r, s\})$ have been output for routing.

(3) All pins reached from $N \cap modules(C)$ are in $pins(C) \cup pins(N \cap U)$.

We now claim that algorithm *find_route*(r, s) obtains a topological routing of the wire (r, s) that preserves the invariant. To establish this, we need to show that

(A) The algorithm actually finds the route between r and s.

(B) The region enclosed by this route and the curve C contains no pins that have not been routed to.

To prove (A), we need to show:

(A1) For each value of $currentpin$, clockwisepin($currentpin$) (i.e., the pin clockwise from $currentpin$) is defined and different from $currentpin$.

(A2) The net $(l, currentpin)$ in step 3 of Figure 13 is already routed, so it is possible to follow this route.

(A3) $currentpin$ does not assume the same value twice.

For (A1), we simply assume that each module has atleast two pins. Modules with a single pin may be ignored initially and routed to after the remaining routes have been made. For (A2), let the value of the *currentpin* and l at the start of the i'th iteration of the **while** loop be c_i and l_i, respectively. We note that $c_1 = s$ and $l_1 \in pins(C)$. If $l_i \in pins(C) \cup pins(N \cap U)$, then from conditions (2) and (3), it follows that $c_{i+1} \in pins(C) \cup pins(N \cap U)$ and wire (l_i, c_{i+1}) has been routed. Suppose there is an $l_i \notin pins(C) \cup pins(N \cap U)$. Let l_j be the first such l_i. Since $j > 1$, l_{j-k} and c_{j-k+1} are in $pins(C) \cup pins(N \cap U)$ for $k \geq 1$. Since c_j and l_j are on the same module, it follows that $module(c_j) \notin N$. So, $l_j \in ext\_pins(C)$ and $c_j \in pins(C)$. From the way algorithm *Testing_Planar_Routability* works, it follows that (l_{j-1}, c_j) is a segment of the curve C and that curve C when oriented from r to s, first reaches l_{j-1} and then c_j via wire (l_{j-1}, c_j). Hence $l_{j-1} \in pins(C)$. Since $l_{j-2} \in pins(C)$ (by assumption on j), $c_{j-1} \in pins(C) \cup pins(N \cap U)$. Further, since c_{j-1} is a module of C, $c_{j-1} \in pins(C)$. Now, since (l_{j-1}, c_j) is a segment of C and c_{j-1} is one pin clockwise from l_{j-1} and a pin of C, it follows that (l_{j-2}, c_{j-1}) is a segment of C oriented from l_{j-2} to c_{j-1}. Continuing in this way, we conclude that (l_1, c_2) is a segment of C oriented from l_1 to c_2. However, we know that when C is oriented from r to s there is only one wire segment that includes a pin of $module(s)$ and this is oriented to $module(s)$. I.e., the orientation is c_2 to l_1, a contradiction. Hence, there is no $l_i \notin pins(C) \cup pins(N \cap U)$. Also, no $l_i \in \{r, s\}$ at the start of a **while** loop iteration. From condition (2), it follows that all encountered (l_i, c_{i+1}) have been routed.

For (A3), suppose that $c_i = c_j$ for some i and j, $i < j$. Since (l_{i-1}, c_i) and (l_{j-1}, c_j) are two-pin nets, it follows that $l_{i-1} = l_{j-1}$. Now, since c_{i-1} and c_{j-1} are, respectively, one pin counterclockwise from l_{i-1} and l_{j-1}, it follows that $c_{i-1} = c_{j-1}$. Continuing in this way, we see that $s = c_1 = c_{j-i+1}$. This implies that net $(l_{j-i}, c_{j-i+1}) = (r, s)$ has already been routed. But, it hasn't. So, no c_i is repeated.

(B) follows from the fact that *find_route* reaches only pins in $pins(C) \cup pins(N \cap U)$, condition (3) and the fact that *find_route* follows existing routes without enclosing any new pins. $\square$

Lemma 5 : *If the algorithm Testing_Planar_Routability (given in Figure 8) terminates in step 4(b), the RI is not planar routable.*

Proof : If the algorithm terminates in step 4(b), then let r and s be as in step 4. r is at the top of stack A and s is in stack B but not at the top. Let x be at the top of stack B and let y be the pin such that $net(y) = net(x)$. y must currently be on stack A as x

can be put on stack B (see step 4(c)) only if y is on stack A. When one pin of a net is in stack A and the other in stack B, the pins can leave the stacks together (step 4(a)) or not at all. Since x is on stack B at termination, y must still be on stack A and hence must be lower than r (as r is at the top). So, there is a curve $y \ldots r$ in the RI. Furthermore, curves $y \ldots r \ldots s$ and $y \ldots r \ldots x$ must exist as this is the only way s and x can get to stack A and then to stack B. Figure 12(a) shows an example curve $y \ldots r \ldots s$. This figure assumes that $module(s) \neq module(r)$. The proof for the case $module(s) = module(r)$ is similar. Let m be the module at which the curves $y \ldots r \ldots s$ and $y \ldots r \ldots x$ diverge (Figure 12(b)). Note that m may be $module(r)$ or a latermodule on the curve $y \ldots r \ldots s$. Let u be the pin of m that is the last pin of m on curve $y \ldots r \ldots s$ and let v be the corresponding pin for $y \ldots r \ldots x$. Since all nets are two-pin nets, $u \neq v$. Since x is above s in stack B, v must be on the curve $y \ldots r \ldots s$. The curve $C = y \ldots r \ldots v \ldots x$ is a closed curve. We see that $r \in pins(C)$, and $s \in ext\_pins(C)$, and $net(s) = net(r)$. So, s and r satisfy the conditions of Lemma 2 and the RI is not planar routable. $\square$

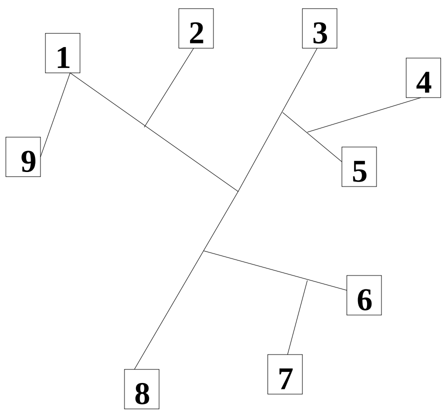
Theorem 1 : *The algorithm Testing_Planar_Routability (given in Figure 8) is correct.*

Proof : Follows from Lemmas 4 and 5. $\square$

The algorithm of Figure 8 is easily implemented to have complexity of $O(n)$ where n is the total number of pins. For this we need to use an array *status[1..n]* to maintain the current status (i.e., on stack A, on stack B, on neither) of each pin.

4 Topological Routability of Multi-pin Nets

We shall refer to the extension of *Testing_Planar_Routability* or TPR to the case where some or all nets may have more than two pins as MTPR. The MTPR problem may be solved in linear time by mapping MTPR instances into graph planarity instances [3, 1]. However, the known linear time algorithms [12] for graph planarity are complex and one is motivated to explore the possibility that simpler algorithms exist for MTPR (just as they do for TPR). Unfortunately, this is not the case. We show, in Theorem 2, that any algorithm for MTPR can be used to test graph planarity with no increase in complexity. In Theorem 3, we show that the problem of determining the maximum number of topologically routable nets of an MTPR instance is NP-hard. For the case where all the pins are two-pin nets, we can use the construction of [1] and the algorithm of [13] to find the maximum subset that is



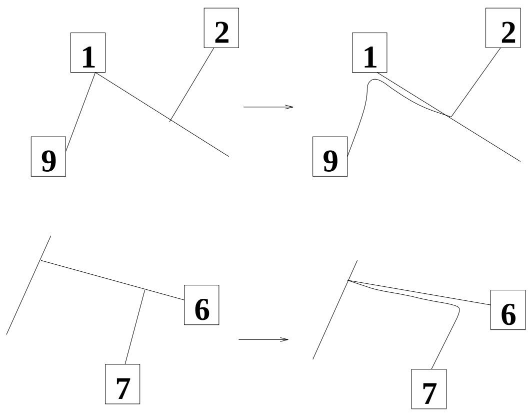


Figure 14: Realization of a planar net using one Steiner point

topologically routable. The complexity of the resulting algorithm is $O(n^2)$, where n is the total number of nets. Theorem 2 motivates the quest for a simple linear time algorithm for a restricted version of MTPR. We show that the algorithm of Figure 8 may be extended to handle MTPR instances in which every pair of modules remains connected (though not neccessarily by a net) when all nets other than two-pin nets are eliminated.

Theorem 2 : *Let I be an instance of graph planarity. I can be transformed in linear time, into an instance I' of MTPR such that I is planar iff I' is topologically routable.*

Proof : From the constructions of [3, 1], it follows that the topological routability of an MTPR instance does not depend on the specific placement of the modules. Hence, in constructing I', we need not specify the module placement. I' is obtained from I by replacing each edge (i, j), $i < j$ of I by a module M_{ij} with two pins M_{ij}^1 and M_{ij}^2. The nets of I' are $N_i = \{M_{ij}^1 \mid i < j\} \cup \{M_{ji}^2 \mid j < i\}$, $1 \leq i \leq n$ where n is the number of vertices in I.

If I' is planar routable, then each net, N_i, has a planar realization that does not contact the realization of any other net. This realization connects the pins of N_i together, possibly using some Steiner points (See Figure 14(a)).

If N_i is a two-pin net, then introduce vertex v_i anywhere on the wire connecting the two pins of N_i. If N_i has more than two pins, then by using the transformations of Figure 14(b), we can reduce the number of Steiner points to one and also ensure that each pin of N_i has exactly one wire connected to it. There transformations preserve the planarity of the routing. The sole surviving Steiner point is replaced by the vertex v_i.

Now each wire that connects to v_i connects to module M_{ij} or M_{ji}. This module has another wire connecting to vertex j. Remove M_{ij} (or M_{ji}) and join the ends of these two wires together by a line joining the terminals of M_{ij} (or M_{ji}). We now have a planar embedding of I.

If I is planar, then start with its planar embedding. Replace v_i by a Steiner point; place M_{ij} anywhere on the embedding of edge (i, j), $i < j$; split the edge (i, j) at M_{ij} and connecting the two ends (at the split point) to the terminals of M_{ij}. This yields a topological routing of I'.

Hence, I is planar *iff* I' is topologically routable. $\square$

Let MTPRmax be the problem of determining whether or not k of the nets of an MTPR instance are topologically routable. To show that MTPRmax is NP-complete, we use the following problem that is known to be NP-complete [14].

Planar Subgraph: Given Graph $G = (V, E)$, and a positive integer $k \leq \mid V \mid$. Is there a subset $V' \subseteq V$ with $\mid V' \mid \geq k$ such that the subgraph induced by the V' vertices is planar?

Theorem 3 : *MTPRmax is NP-complete.*

Proof : It is easy to see that MTPRmax is in NP. Also, from an instance I of the planar subgraph problem, we can construct an instance I' of MTPRmax by replacing edges by modules as in Theorem 2. It is easy to see that I' has k nets that are topologically routable *iff* I has an induced subgraph with k vertices that is planar. $\square$

Any instance I of MTPR may be transformed into an instance I' of TPR which includes unordered modules (i.e., modules whose terminals may be rearranged at will). I' has the property that there is an arrangement of terminals for each of the unordered modules which results in I' being topologically routable *iff* I is topologically routable. To obtain I' from I, for each multi-pin net N_i of size k, $k > 2$, we introduce an unordered module UM_i with k pins. The net N_i is replaced by k two-pin nets, one pin of each of these nets is an original pin of N_i and the other a pin of UM_i. Since planar routability is not affected by module placement, UM_i may be placed anywhere in the routing region. An example is given in Figure 15.

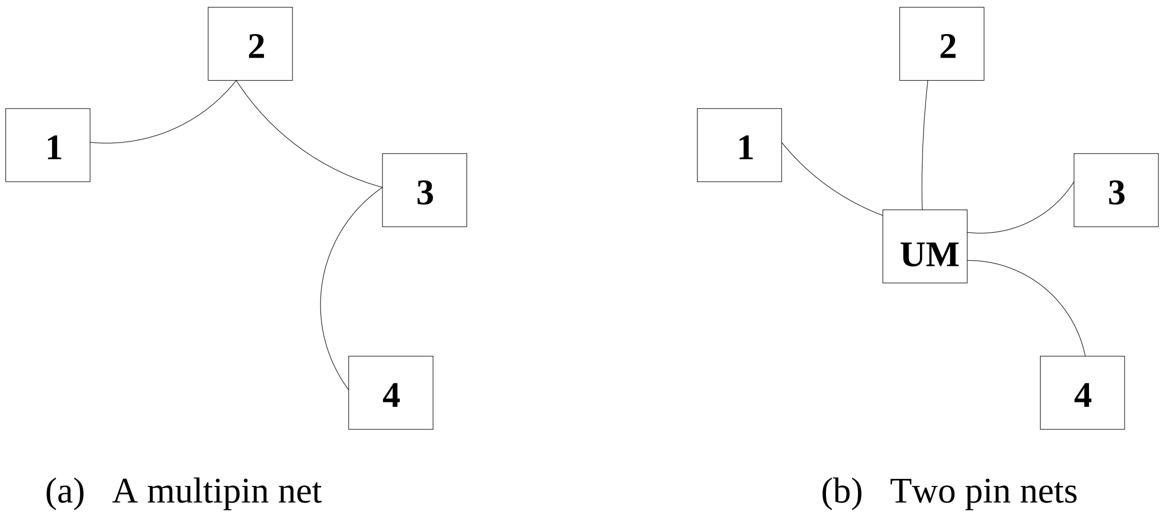


Figure 15: Transformation from multi-pin to two-pin nets

Theorem 4 : *The pins of each unordered module of I' can be ordered so that the resulting instance of TPR is topologically routable iff I is topologically routable.*

Proof : If the pins of the UM_i's can be so ordered, then a topological routing of I is easily obtained from the topological routing of I' (simply replace each UM_i by a Steiner point). If I is topologically routable, then using transformations similar to those in Figure 15(b), we may transform the topological routing into one in which each multi-pin net N_i of size $k > 2$ is routed using exactly one Steiner point. This Steiner point is replaced by module UM_i and the pin ordering is determined by the topological routing around the Steiner point. $\square$

If we knew which terminal orderings of the UM_i's to use, we could simply convert each UM_i to an ordered module and run the TPR algorithm. Unfortunately, we do not know this. Therefore we need to modify algorithm *Testing_Planar_Routability* so as to properly handle unordered modules. As in section 3, we may assume that I is a single component. For our modification to work, we assume that I remains a single component when all multi-pin nets N_i of size $k > 2$ are eliminated from I. The modified algorithm MTPR is given in Figure 16.

Lemma 6 : *If the algorithm MTPR halts in step 4, then the wires are planar routable.*

Algorithm MTPR

```
Step 1: For each multi-pin net $N_i$ of size $k > 2$, introduce a new unordered module $UM_i$
    with $k$ pins and replace $N_i$ by $k$ two terminal nets as described earlier.

Step 2: Let $m$ be any ordered module and let $p$ be any pin of $m$ which corresponds to an
    original two-pin net.

Step 3: Examine the pins of $m$ in counterclockwise order beginning at pin $p$. When a pin
    $q$ is being examined compare $net(q)$ and $net(r)$ where $r$ is the pin (if any) at the top
    of stack $A$. If stack $A$ is empty or $net(q) \neq net(r)$ then add $q$ and the remaining pins
    of $m$ to the top of stack $A$. Otherwise output $(q, r)$ and unstack $r$ from $A$.

Step 4: If both stacks $A$ and $B$ are empty, then terminate.

Step 5: Let $r$ be the pin at the top of stack $A$. Let $s$ be the pin such that $net(r) = net(s)$.
    If $module(s)$ is an unordered module then go to step 6.

    (a) If $s$ is at the top of the stack $B$, then [output $(r, s)$; unstack $r$ from $A$ and $s$
        from $B$; go to start of Step 4].
    (b) If $s$ is in stack $B$ but not at the top, then [output("The RI is not planar
        routable"). Terminate].
    (c) If $s$ is in stack $A$, then [unstack $r$ from A; add $r$ to stack $B$; go to the start of
        Step 5].
    (d) If $s$ is in neither of the stacks then [ set $p$ to $s$; let $m$ be the module containing
        $s$; go to Step 3].

Step 6:

    (a) If $module(s)$ is at the top of stack $B$, then [output $(r, s)$; unstack $r$ from $A$;
        mark pin $s$ as having been seen. If all pins of $module(s)$ have been marked then
        unstack $module(s)$ from $B$; go to start of step 4]
    (b) If $module(s)$ is on $B$ but not at the top, then [output("The RI is not planar
        routable"). Terminate].
    (c) If $module(s)$ is not in stack $B$, then [unstack $r$ from $A$; mark pin $s$ as having
        been seen; add $module(s)$ to the top of stack $B$; go to start of step 5].
```

Figure 16: Topological routing of multi-pin net for restricted version

Proof : The proof is very similar to that of Lemma 4. The same invariant holds. When we put an unordered module on stack B and mark a pin (if that is the first pin marked) then we connect this pin to the unordered module. See that there is no enclosed region and the invariant holds true. Now, if we are routing another pin (of multi-pin net) then the pin it has to be connected is on the unordered module. So as soon as we reach the unordered module, the next pin is chosen as the pin it has to be connected to (this also defines the order of pins in the unordered module). The proofs apply in this case as we have made all nets two-pin nets. □

Lemma 7 : *Let I be an RI that contains a curve $C = P_1P_2 \ldots P_j$. Let $R = R_1R_2 \ldots R_k$ and $S = S_1S_2 \ldots S_l$ be two curves such that $module(R_1) = module(P_d)$ for some d, $1 \leq d \leq j$ and $first(R_1) \in ext\_pins(C)$ and $module(S_1) = module(P_e)$ for some e, $1 \leq e \leq j$ and $first(S_1) \in pins(C)$.*

Let C be such that $first(P_1)$ and $last(P_j)$ are part of the same net N. Assume that there exist two pins a and b such that $a \in pins(C) \cup pins(S)$, $b \in ext\_pins(C) \cup pins(R)$ and $net(a) = net(b) \neq N$.

I is not planar routable.

Proof : Follows from Lemma 1. □

Lemma 8 : *If algorithm MTPR terminates in steps 5(b) or 6(b), the RI is not planar routable.*

Proof : Suppose the algorithm terminates in step 5(b). Let r and s be as in step 5 and let x be at the top of stack B. Note that r and s define a two-pin net. If x is a two-pin net, then the RI is not planar routable (see proof of Lemma 4). So assume that x is an unordered module (note that only pins of two-pin nets and unordered modules get on to stack B). Module x must have atleast one marked and one unmarked pin. Let C be the curve defined by the stack A segment from r to s when s was at the top of stack A just prior to being transferred to stack B. From the working of MTPR, it follows that there is a pin $p \in pins(C)$ from which a path was traced to the multi-pin net corresponding to module x. Furthermore, there is atleast one pin a of the multi-pin net that is on a path from a pin that is not in pins(C). A possible situation is shown in Figure 17. The conditions of Lemma 7 are satisfied and the RI is not planar routable.

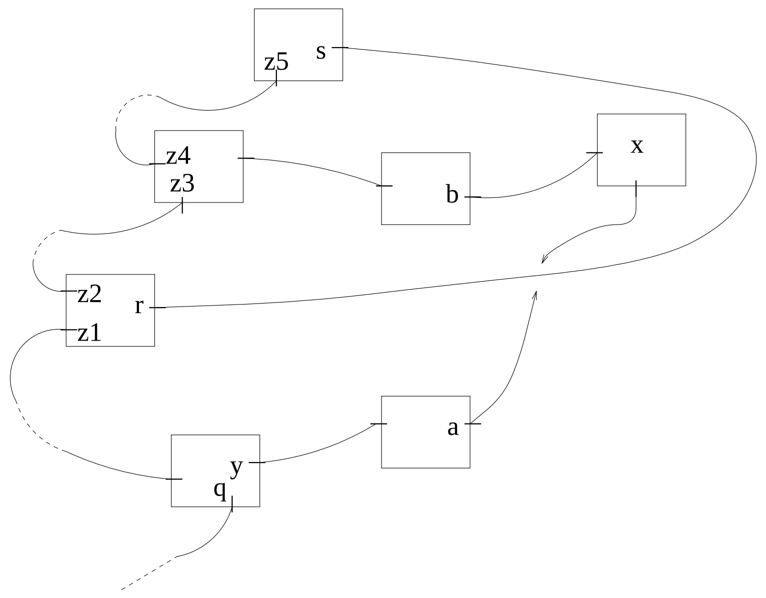


Figure 17: An possible situation where the RI is unroutable

If the algorithm terminates in step 6(b), then r is a pin of a multi-pin net and $module(s)$ is an unordered module. Let x be at the top of stack B at the time of termination. Let j be one of the pins that have already been routed to $module(s)$. Let C be the curve defined by the stack A segment at the time pin j was output for routing. Let S be the curve or pin in $pins(C)$ that was used to reach pin x. Let y be such that $net(x) = net(y)$. Since y must be below r on stack A and r is a net of a multi-pin net, the path from y to r on stack A must include a pin in $ext\_pins(C)$. By setting a and b of Lemma 7 to x and y respectively, we see that the conditions of Lemma 7 are satisfied and the RI is unroutable. The proof for the case x is an unordered module is similar. $\Box$

5 Implementation of Two-Pin Algorithm

While the correctness proof for our algorithm is somewhat involved, the algorithm itself is quite simple and easy to implement. To get good performance we implemented stack A as a stack of modules rather than one of pins as described in Section 3. So, when step 2 of Figure 8 adds q and the remaining pins of m to stack A, we simply add a record of the type (m, q, l) where l is the last pin of m to the stack. Also, to get the top pin of stack A,

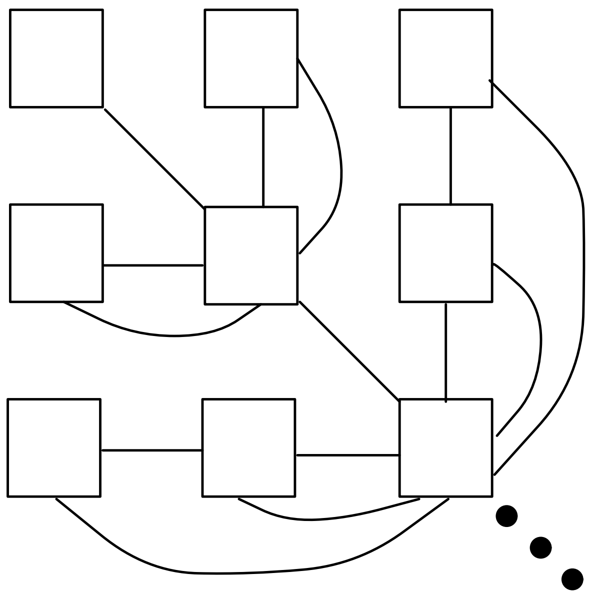


Figure 18: Tree-like Connected Circuits

we look at the top record (m, q, l). The top pin is l. To delete this pin, the top record is changed to $(m, q, p(l))$ where $p(l)$ is the predecessor of pin l unless $q = l$. In the latter case, the record (m, q, l) is deleted from the stack. The role of array *status* needs to be changed to support this change in stack structure. We now keep a *status* for a module as well as for a pin. A module's *status* reflects whether or not it is in stack A and a pin's *status* reflects whether or not it is in stack B.

The algorithm of [1] is a two step algorithm:

Step 1:Merge modules together to obtain an equivalent routing problem in which all pins are on the periphery of a routing region.

Step 2:Determine the feasibility of the equivalent problem using a single stack scheme.

To implement step 1, we performed a traversal of the modules. Each module was represented as a singly linked circular list of pins. With this representation, modules can be merged efficiently. By contrast, for the algorithm of Figure 8, modules were represented using doubly linked circular lists.

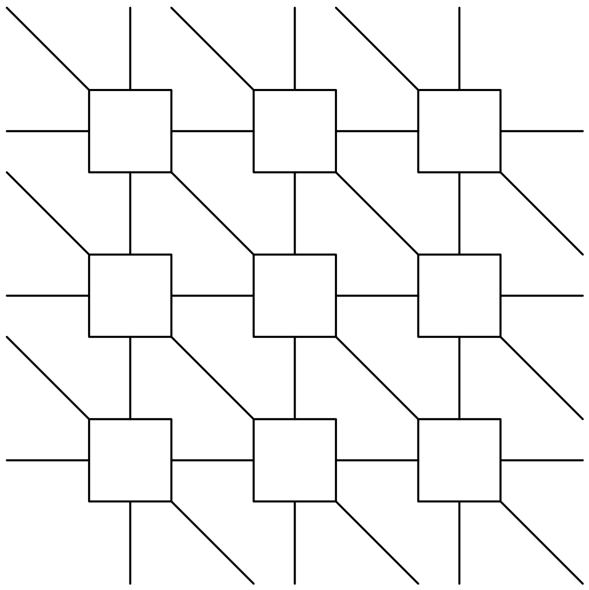


Figure 19: Six-way Connected Circuits

6 Experimental Results

We implemented our algorithm for two-pin nets and that of [1] in C and obtained execution times using both circuits that are routable and those that are not. The routable circuits used are highly structured ones as shown in Figures 18 and 19 as well as randomly generated ones. The nonroutable circuits used were obtained by modifying the tree-like circuits of Figure 18.

NP	864	704	1440	6048	6944	24928
NM	16	25	49	100	225	400
Our	3.40	2.91	5.53	22.90	27.40	93.30
[1]	5.45	4.36	9.10	37.80	44.90	162.80

NP = Number of pins in the circuit

NM = Number of modules in the circuit

Table 1: Tree-like Connected Circuits

NP	1792	1920	522	8352	9856	17396
NM	25	49	100	100	225	400
Our	8.48	10.17	4.20	44.54	53.60	105.40
[1]	11.08	11.93	9.52	55.18	66.25	121.90

Table 2: Six-Way Connected Circuits

NP	92	1472	2944	11776
NM	28	28	28	28
Our	0.91	6.54	12.76	49.95
[1]	0.98	9.18	18.17	78.20

Table 3: Random Circuit

The timing results for the routable circuits are shown in Tables 1, 2, and 3, respectively. The times are in milliseconds and the programs were run on a SUN 4 workstation. on tree-like circuits, the algorithm of [1] took 65% more time than ours, on average; on six-way circuits, it took approximately 40% more time; and on random circuits, it took approximately 37% more time.

Our algorithm has a distinct advantage over that of [1] when working with nonroutable circuits. The algorithm of [1] must complete its step 1 before it can detect infeasibility, whereas our algorithm can detect infeasibility at any stage. Hence, it is possible for our algorithm to take much less time than that of [1] when working on such circuits. The results from three test circuits is given in Table 4. The algorithm of [1] took approximately 3 to 5 times as much time as did ours.

NP	868	6944	13888
NM	25	225	225
Our	3.70	16.24	30.30
[1]	15.50	45.70	92.50

Table 4: Faster Termination for NonRoutable Circuits

7 Conclusion

We have developed a relatively simple and fast linear time algorithm to test the planar topological routability of a collection of two-pin nets. The algorithm is faster than the linear time algorithm of [1]. We have also shown the equivalence of the multi-pin net routability problem and the graph planarity problem. This implies that we cannot solve the multi-pin problem by a simpler algorithm than used for graph planarity. Our simple two-pin algorithm can be extended to work on routing instances that remain connected following the elimination of all multi-pin nets. These are the same instances that can be solved by the specialized algorithm of [1]. We also show that determining the maximum number of routable nets is NP-hard when some or all of the nets have more than two pins. When all nets are two-pin nets, this can be done in $O(n^2)$ time.

References

[1] M. Marek-Sadowska and T. Tarng, "Single-Layer Routing for VLSI: Analysis and Algorithms," *IEEE transactions on Computer-Aided Design*, vol. CAD-2, no. 4, 1983.

[2] C. Hsu, "General River Routing Algorithm," in *ACM/IEEE Design Automation Conference*, pp. 578–583, 1983.

[3] C. Leiserson and R. Pinter, "Optimal placement for river routing," *SIAM Journal on Computing*, no. 12, pp. 447–462, 1983.

[4] A. Lim, S. Cheng, and S. Sahni, "Optimal Joining of Compacted Cells," *IEEE Transactions on Computer*, accepted in 1991. Extended abstract in 1992 Brown/MIT Advanced Research in VLSI and Parallel Systems.

[5] A. Mirzaian, "River Routing in VLSI," *Journal of Computer and System Sciences*, no. 34, pp. 43–54, 1987.

[6] A. Mirzaian, "A Minimum Separation Algorithm for River Routing with Bounded Number of Jogs," in *International Conference in Computer-Aided Design*, pp. 10–13, 1989.

[7] R. Pinter, "On routing two point nets across a channel," in *ACM/IEEE Design Automation Conference*, pp. 899–902, 1982.

[8] R. Pinter, "River-Routing: Methodology and Analysis," in *Third Caltech Conference on VLSI*, March 1983.

[9] A. Siegel and D. Dolev, "The separation for general single layer wiring barriers," in *VLSI Systems and Computations*, pp. 143–152, 1981.

[10] T. Tuan and S. Hakimi, "River Routing with Small Number of Jogs," *SIAM J. Discrete Math.*, vol. 3, no. 4, pp. 585–597, 1990.

[11] T. Tuan and K. Teo, "On River Routing with Minimum Number of Jogs," *IEEE transactions on Computer-Aided Design*, vol. 10, no. 2, pp. 270–273, 1991.

[12] John Hopcroft and Robert Tarjan, "Efficient Planarity Testing," *J. ACM*, vol. 21, no. 4, pp. 549–568, 1974.

[13] K. J. Supowit, "Finding a Maximum Planar Subset of a Set of Nets in a Channel," *IEEE transactions on Computer-Aided Design*, vol. CAD-6, no. 1, 1987.

[14] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP- Completeness.* W.H. Freeman and Company, 1979.