Routing in a Rectangle with k-ary Overlap

Jon Hamkins*

Beckman Institute
University of Illinois
405 N. Mathews Ave.
Urbana, IL 61801
hamkins@grinch.cc.uiuc.edu

Donna J. Brown*

Coordinated Science Lab.
University of Illinois
1101 W. Springfield Ave.
Urbana, IL 61801
djb@uicsl.csl.uiuc.edu

Abstract

A layout for the k-ary overlap rectangle routing problem (k-RRP) is a set of paths connecting pairs of terminals on the boundary such that each edge hosts at most k distinct nets. We give necessary and sufficient conditions for the existence of such a layout, and present an algorithm that constructs a layout, if one exists, in time $O(kN)$, where N is the number of gridpoints in the rectangle. Extensions to more general problems are also discussed.

1 Introduction

Recent technological advances are making multilayer VLSI a reality. As a consequence there is increasing interest in the development of CAD tools which support multilayer routing. We consider here the well-known problem of routing within a rectangle. Corresponding to multiple routing layers, we assume that wires can overlap (i.e., run on top of each other). Specifically we consider the problem of constructing a "layout"; that is, we determine the path of each wire through a 2-dimensional rectangular grid, but we do not specify on which routing layer each path segment lies. This separation of a problem into layout vs. layer assignment subproblems has been seen a number of times in the literature for no-overlap wiring models (see, e.g., [MP86], [GK87]). However, this is the first work we know of which specifically addresses the switchbox layout problem where full k-ary overlap is allowed. We present an algorithm which, given a k-ary rectangle routing problem, constructs a k-ary overlap layout, whenever one exists. We also give necessary and sufficient conditions for the existence of such a layout.

The well-known *Rectangle Routing Problem*, or *RRP*, is a disjoint paths problem. We are given a *rectangle* consisting of n columns, numbered 1 to n from left to right, and m rows, numbered 1 to m from bottom to top. *Terminals*, represented by numbers, are located at gridpoints on the boundary of the rectangle. Each boundary gridpoint can have at most one terminal, except that a corner gridpoint may have up to two terminals. Terminals with the same number form a *net*. A net with two terminals is a *two-terminal net*, and a net with more than two terminals is a *multiterminal net*. A solution to a RRP is given by edge-disjoint paths in the rectangle which connect each net's terminals together; this solution is called a *layout*.

The first rigorous characterization of RRP's was given by Frank [Fra82]. Elaborating on the multicommodity flow ideas of Okamura and Seymour [OS81], [Fra82] considers two-terminal nets, specifies the conditions for the existence of a layout, and provides a layout if one exists. Figure 1a illustrates a RRP for which the rectangle consists of $n = 7$ columns and $m = 5$ rows. There are ten two-terminal nets to be routed. A solution is given in Figure 1b, where the layout produced consists of ten edge-disjoint paths, one path for each net. Frank's algorithm runs in $O(nm)$ time and can have as many as $O(nm)$ knock-knees. Mehlhorn and Preparata [MP86] improve Frank's algorithm by bounding the number of knock-knees to $O(b)$, thereby achieving a running time of $O(b \log b)$, where b is the number of gridpoints on the boundary.

In this paper, we consider a more general version of the RRP, called the *k-ary overlap RRP*, or k-RRP.

*This work was supported in part by the Joint Services Electronics Program under Contract N00014-90-J-1270.

0-8186-2610-0/92 $3.00 © 1992 IEEE

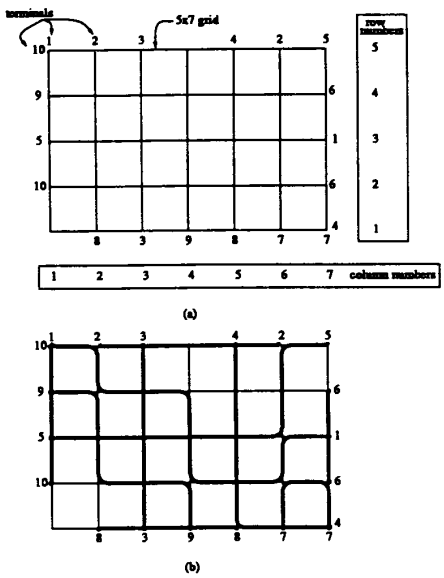

Figure 1. (a) A (classical) 1-RRP and (b) a solution.

Instead of requiring paths to be edge-disjoint, we allow up to k-ary overlap. That is, any edge may host a path-segment from at most k different nets. Correspondingly, each gridpoint on the boundary can be the terminal of up to k nets, except that a corner point may be the terminal of up to $2k$ nets. It is easy to see that the k-RRP is identical to the RRP when $k = 1$. Figure 3a shows a 2-RRP consisting of 18 two-terminal nets. A routing for this problem is given in Figure 3b; note that each edge can be occupied by 0, 1, or 2 nets.

Others have considered a different setting than the k-RRP. For example, [Fra85] and [BM86], in independent work, present algorithms for routing without overlap in an arbitrary planar graph. Their work is more general in that the routing domain need not be a rectangular grid, but more restrictive in that no overlap is allowed. However, their work may be applied to the k-ary overlap problem: a k-RRP may be recast as a no-overlap problem in a planar graph. Unfortunately, when extended to the k-RRP problem, [Fra85] has a running time of $O(n^3m^3 \log nm)$, and [BM86] produces a running time of $O(kn^2m^2)$. In this paper, we specifically solve the k-ary overlap RRP and do so with a running time of only of $O(knm)$, which compares directly to the running time obtained in the no-overlap RRP algorithm presented in [Fra82].

Before we discuss our results, it is worth noting that the problem is not as simple as it may appear. In particular, we cannot obtain a k-ary overlap layout by simply expanding the size of the rectangle k times in each dimension, solving the no-overlap problem, and then shrinking the layout to obtain the k-ary overlap layout. The difficulty is that it is not at all clear how the terminals should be placed onto the expanded rectangle. If placed incorrectly, the resulting no-overlap problem may not be routable even though the smaller rectangle does have a k-ary overlap layout. See Figure 2. Although some choice of terminal placement may admit a no-overlap layout, the terminal placement algorithm would have to minimize the density (defined below) of every row and column. There could be over $(2n + 2m)^{k!}$ different choices for where to reasonably place the terminals on the expanded problem, leading to an unacceptably large running time. Furthermore, we do not know whether every solvable $n \times m$ k-RRP even has a corresponding solvable $kn \times km$ 1-RRP. That is, even if all $(2n + 2m)^{k!}$ corresponding 1-RRP's are not routable, it is not known whether this implies the k-RRP is unroutable.

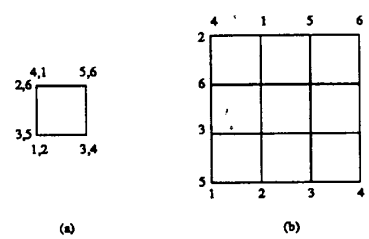

Figure 2. (a) A solvable 2-RRP and (b) a corresponding, but unsolvable 1-RRP.

2 Definitions

In order to precisely state our results, we need some definitions. Following the terminology in [MP86], a *vertical cut* (*v-cut*) runs vertically between two columns, c and $c + 1$, and is denoted by $(c, c + 1)$, and a *horizontal cut* (*h-cut*) runs horizontally between two rows, r and $r + 1$, and is denoted by $(r, r + 1)$. The *capacity* of a cut is k times the number of grid edges split by the cut, and represents the maximum number of nets which could possibly cross that cut. The *density* of a cut is the number of nets which have a terminal on each side of the cut, and therefore must cross the cut. For instance, in Figure 3a, it is easy to see that h-cut $(2, 3)$ has density 12 and capacity 12, and that v-cut (2,3) has density 7 and capacity

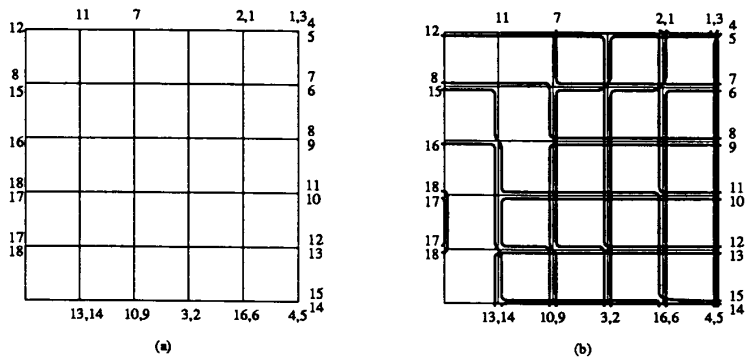

Figure 3. (a) A 2-RRP and (b) a solution.

12. The *k-ary column criterion* states that the density of any v-cut is no greater than its capacity. The *k-ary row criterion* is analogous. Unfortunately, as [Fra82] points out, even for the case $k = 1$ these criteria alone are not sufficient for the existence of a layout, so stronger criteria are needed.

Suppose a v-cut or h-cut has density d and capacity C. We say the cut is *near-k-saturated* if $0 < C - d < k$, *k-saturated* if $C = d$, and *over-k-saturated* if $C < d$. The *extended degree* of a gridpoint is k times the degree of the gridpoint plus the number of terminals at the gridpoint. A routing problem is called *standard* if each boundary gridpoint has even extended degree. Suppose a k-RRP has t h-cuts that are k-saturated, and consider any v-cut $(c, c+1)$. This gives rise to $t+1$ regions to the left of the cut: $T_1, ..., T_{t+1}$. We say T_i is an *odd set* if k times the number of edges leaving region T_i plus the number of nets with exactly one terminal in T_i is odd. The number of such odd T_i's is called the *parity congestion* of v-cut $(c, c+1)$. (Note that the parity congestion of a v-cut would be the same if we defined the $t+1$ regions to be to the right of the v-cut.) The *revised k-ary column criterion* states that the density *plus* the parity congestion of any v-cut does not exceed its capacity. The *revised k-ary row criterion* is defined analogously.

For instance, consider the 1-RRP in Figure 4a. Since no v-cut or h-cut is over-1-saturated, the 1-ary row and column criteria are satisfied. However, the revised row and column criteria are *not* satisfied, and the problem is unroutable. As shown, it has three 1-saturated h-cuts. In conjunction with the arbitrary v-cut shown, these give rise to the four regions T_1, T_2, T_3, and T_4. T_1 has six grid edges leaving the region and three nets with exactly one terminal inside; since 6+3

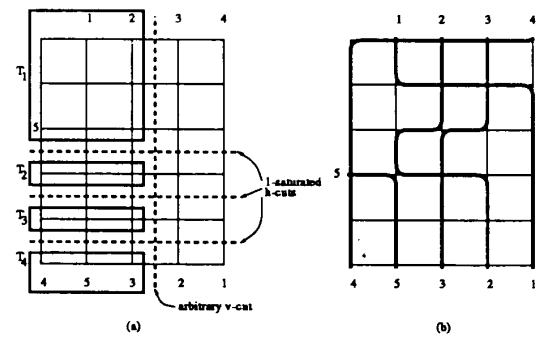

Figure 4. [Fra82] (a) An unroutable 1-RRP. (b) A routable 1-RRP.

is odd, T_1 is odd. Similarly, T_2, T_3, and T_4 are odd, so the parity congestion of the indicated v-cut is 4. Since this cut also has density equal to 4, its density plus parity congestion is 8, which exceeds its capacity of 6. Thus, the revised 1-ary column criterion does not hold, and the given problem is not routable. It is interesting to note, however, that the nearly identical 1-RRP in Figure 4b is routable.

3 The Algorithm

In this section we motivate our approach for constructing a layout for a k-RRP whenever one exists, and present an $O(knm)$ time algorithm to do so. The following section provides a proof of its correctness and time complexity.

Our algorithm proceeds in a row-by-row fashion, starting with the top row. After the routing for a row

has been completed, the routing for the row just below it is begun. We perform the routing of each row left to right. Once the algorithm has routed some portion of the rectangle, it will not alter this routing in subsequent steps. Hence, as the algorithm proceeds, a smaller problem remains to be solved. This remaining problem has the shape of a rectangle, except that a left portion of the top row is missing. (This missing portion represents that part of the row which has already been routed.) We call this shape a *near-rectangle.* (See Figure 5.) There are terminals specified along the bottom, left, and right boundaries of the near-rectangle, and routed nets which enter from the top. Referring to the near-rectangle in Figure 5, we note that nets entering from the top are viewed as having terminals at the corresponding top gridpoints. For instance, terminals may be located at P and P', but not at Q. The *k-ary near-rectangle routing problem*, or k-NRRP, is the problem of constructing a k-ary overlap layout for such a near-rectangle.

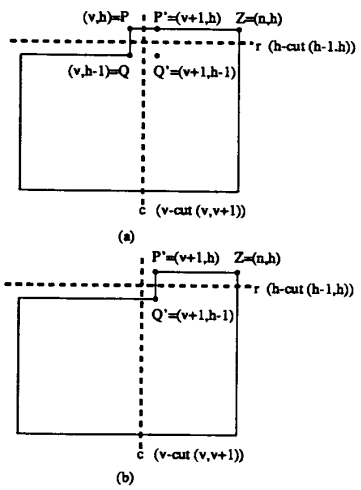

Figure 5. (a) The near-rectangle before P is processed. (b) The near-rectangle after P is processed.

Since it is easier to determine the k-ary row and column criteria than the revised k-ary row and column criteria, the first step of our algorithm will be to transform the given problem into a problem for which these two sets of criteria are equivalent. This involves two lemmas.

Lemma 1 *The k-ary row and column criteria and the revised k-ary row and column criteria are equivalent for a standard k-NRRP.*

Proof. If a set T_i is odd, then it contains an odd number of gridpoints of odd extended degree. By the definition of standard, every boundary gridpoint in a set T_i has even extended degree. Every nonboundary gridpoint also has even extended degree, so T_i is not odd. T_i was arbitrary, so in fact the parity congestion of any v-cut or h-cut is zero. Thus, the k-ary column criterion is equivalent to the revised k-ary column criterion. The same argument holds for the row criterion. □

Lemma 2 *Any k-RRP satisfying the revised k-ary row and column criteria can be converted, in $O(knm)$ time, to a standard k-RRP which satisfies the k-ary row and column criteria, and the solution to the standard k-RRP easily yields the solution to the original k-RRP.*

Proof. We identify those regions of the rectangle bounded by a k-saturated row or column. Within each region, we pair all boundary gridpoints having odd extended degree, in a clockwise fashion, and consider these pairs to be additional nets to be routed. The resulting problem is a standard k-RRP which satisfies the k-ary row and column criteria. Several authors provide the details of this proof (e.g. [MP86]), so we omit it here. □

Our algorithm will take any solvable k-RRP, transform it to a standard k-RRP, which is also solvable, and then process each gridpoint one at a time. By Lemmas 1 and 2, the algorithm will perform correctly provided that after each gridpoint is processed, the problem remaining still satisfies the k-ary row and column criteria, and is still standard. We do this by employing a greedy algorithm: take action at the current step to avoid an over-k-saturation problem during the following step.

The current step of the algorithm is the processing of gridpoint P, which has coordinates (v, h), as shown in Figure 5. We must decide which of the nets with terminals at P to route down along PQ and which to route right along PP'. Additionally, it may be necessary to route some nets along QPP' (i.e., from Q to P to P').

Let c be the v-cut $(v, v+1)$ and r be the h-cut $(h-1, h)$. If neither c nor r is near-k-saturated, then we route the nets with a terminal at P, and no nets route along QPP'. Let η_c be the set of all nets having a terminal at P and which are split by c but not by r, let η_r be the set of all nets having a terminal at P and which are split by r but not by c, and let η_{cr} be the set of all nets having a terminal at P and which are split by both c and r. Let N_c, N_r, and N_{cr} be the

number of nets in η_c, η_r, and η_{cr}, respectively, and let $N = N_c + N_r + N_{cr}$. Of the N nets with a terminal at P, we route the nets of η_r down along PQ and the nets of η_c right along PP'. (Note that if $N_c > k$ then only k of the nets in η_c can be routed along PP', and the remaining must be routed along PQ. Similarly, if $N_r > k$, then not all nets of η_r may be routed along PQ.) The nets of η_{cr} can be routed either way: we route a of them across PP', and the other $N_{cr} - a$ down PQ. See Figure 6. The algorithm is based on this type of intuition; we route each net in the direction that it ultimately must go.

If either r or c is near-k-saturated, the above method won't work, because there is no guarantee that a sufficient number of nets will be routed across c or r to avoid over-k-saturation during the next step. It is still best to route the η_r nets along PQ and the η_c nets along PP', so we do so. We let α_c be the minimum additional number of nets (beyond N_c) that must be routed across c to avoid over-k-saturation during the following step, and let α_r be the minimum additional number of nets (beyond N_r) that must be routed across r. For example, if c is k-saturated and $N_c = 2$, then $\alpha_c = k - 2$. (In addition to the two nets in η_c, $k - 2$ other nets contributing to the density of c must be routed on PP'.) The algorithm's goal is to find a total of at least $\alpha_r + N_r$ nets contributing to the density of r to route along PP', and a total of at least $\alpha_c + N_c$ nets contributing to the density of c to route along PQ. If this is done, then the problem will not be over-k-saturated during the following step.

If routing the nets at P alone is sufficient to avoid over-k-saturation for the following step, then we simply proceed to the next step and process the next gridpoint, P'. If not, we "pull" nets toward P. By "pull", we mean the act of replacing a net having terminals s and t with two nets, one having terminals s and Q, the other having terminals P' and t, and joining the two nets by routing along QPP'. We let p be the minimum number of pulled nets needed to avoid over-k-saturation during the next step.

After processing P, we delete it and its (two) adjoining edges. If this remaining problem is not standard, we create a fictional net with terminals at P' and Q so that once again these two gridpoints, as well as all other gridpoints, have even extended degree.

We now present the detailed procedure for processing gridpoint P:

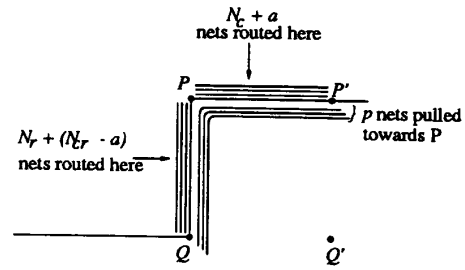

Figure 6. The processing of gridpoint P, assuming $N_c \leq k$ and $N_r \leq k$.

procedure PROCESS(P)
if $N_c > k$ **then**
  route k of the η_c nets along PP';
  route remaining nets ending at P along PQ;
if $N_r > k$ **then**
  route k of the η_r nets along PQ;
  route remaining nets ending at P along PP';
if $N_r \leq k$ and $N_c \leq k$ **then**

$$a := \begin{cases} \frac{1}{2}(N_{cr} + \alpha_c - \alpha_r) & \text{if } \alpha_c > 0 \text{ and } \alpha_r > 0 \\ N_{cr} & \text{if } \alpha_r = 0 \\ 0 & \text{if } \alpha_c = 0 \text{ and } \alpha_r > 0 \end{cases}$$

  if $a > k - N_c$ **then** $a := k - N_c$;
  if $a < N_{cr} - (k - N_r)$ **then** $a := N_{cr} - (k - N_r)$;
  if $a > N_{cr}$ **then** $a := N_{cr}$;
  if $a < 0$ **then** $a := 0$;
  route along PP':
    all nets belonging to η_c;
    a nets belonging to η_{cr};
  route along PQ:
    all nets belonging to η_r;
    $(N_{cr} - a)$ nets belonging to η_{cr};
  $p := \max(0, \alpha_c - a, \alpha_r - (N_{cr} - a))$;
  if $p = \alpha_c - a$
    then choose p nets which cross c and which have a terminal in maximal row;
    else of the nets which have a terminal on $P'Z$ and cross r, choose those p nets which have a terminal furthest to the left;
  for each net i of the p nets chosen, **do**
    let s and t represent the locations of the leftmost and rightmost terminals, respectively, of i;
    replace net i with i' and i'', where the terminals of i' are s and Q and the terminals of i'' are P' and t;
    connect i' to i'' by routing along QPP';
delete P and edges PQ and PP';
if the number of edges at P' (and hence at Q) is odd
  then create a new net with terminals at Q and P';

4 Results

We can now state and prove the following extension of Frank's [Fra82] result:

Theorem 1 *Given a k-RRP, there exists a layout with k-ary overlap iff the revised k-ary row and column criteria hold. Moreover, such a layout, if one exists, can be constructed in time $O(knm)$.*

Proof. ($\Rightarrow$) (Follows the proof in [Fra82].) If a set T_i is odd, then in any solution at least one edge leaving T_i will not be used fully, i.e. at least one edge will have fewer than k nets routed on it. Edges leaving T_i cross either a k-saturated h-cut or the arbitrary v-cut used to define T_i. The edge which is not fully used must cross the v-cut, for otherwise it would have to cross a k-saturated h-cut, which is impossible. Hence, the density of the v-cut plus its parity congestion (the number of odd sets T_i) can be at most the capacity of the v-cut. That is, the k-ary revised column criterion holds. Similarly, the k-ary revised row criterion holds.

($\Leftarrow$) The algorithm given above must provide a k-ary overlap layout whenever the revised k-ary row and column criteria hold. Lemmas 1 and 2 imply that we need only show that after each gridpoint is processed, the remaining problem satisfies the k-ary row and column criteria, and is still standard. We prove these separately, followed by a proof of the time complexity. Let G be the near-rectangle before the processing of P, and let G' be the near-rectangle remaining after P has been processed.

Claim. *G' has no over-k-saturated row or column.*

Proof. Case 1. $N_c > k$. Let e be the number of edges split by r, and let $d_G(r)$ and $d_{G'}(r)$ denote the density of r in G and G', respectively. The maximum number of terminals on PZ is $ke + N$. Since there are at least $2N_c$ terminals on PZ which belong to nets not crossing r, we have $2N_c + d_G(r) \leq ke + N$, or $d_G(r) + (N_c - k) - N_r - N_{cr} \leq k(e-1)$. But the left-hand side of the inequality is $d_{G'}(r)$, and so r is not over-k-saturated in G'. Clearly, c is not over-k-saturated in G', for all k nets routed along PQ contributed to the density of c. The capacity and density of any other v-cut or h-cut remains unchanged, and so no other v-cut or h-cut is over-k-saturated in G'.

Case 2. $N_r > k$. If $v > 1$, the density of v-cut $(v-1, v)$ is the same in G and G', and at least that of v-cut $(v, v+1) = c$ in G'. If $v = 1$, then an argument similar to Case 1 holds. Hence, c is not over-k-saturated in G'. Also, r is not over-k-saturated in G', for all k nets routed along PP' contributed to the density of c. The capacity and density of any other v-cut or h-cut remains unchanged, and so no other v-cut or h-cut is over-k-saturated in G'.

Case 3. $N_c \leq k$ and $N_r \leq k$.

Subcase 3.1. $p = 0$. Note that $\alpha_c - a \leq 0$, so that when $N_c + a$ nets are routed on PP', it meets or exceeds the $N_c + \alpha_c$ nets necessary to avoid over-k-saturation of c in G'. Similarly, $\alpha_r - (N_{cr} - a) \leq 0$, so that when $N_r + (N_{cr} - a)$ nets are routed on PQ, it meets or exceeds the $N_r + \alpha_r$ nets necessary. Again, the capacities and densities of other v-cuts and h-cuts are not affected.

Subcase 3.2. $p = \alpha_c - a$. We see that c is not over-k-saturated in G', for $N_c + a + p = N_c + \alpha_c$ nets are routed on PP', which is the amount necessary. On the other hand, the number of nets routed on PQ is $N_r + (N_{cr} - a) + p \geq N_r + (N_{cr} - a) + \alpha_r - (N_{cr} - a) = N_r + \alpha_r$ which is the amount necessary for r to avoid over-k-saturation in G'. No other v-cut other than c can be over-k-saturated in G', since any other v-cut has the same capacity and density before and after execution of the current step. The only h-cut which could become over-k-saturated is an h-cut $J = (j, j+1)$, that is above both terminals of at least one net chosen by the algorithm, for no other row has its density increased. Use the following counting argument to show it does not become over-k-saturated. Refer to Figure 7 for notation. Let $d_G(J_r)$ be the number of nets which cross J and which have a terminal in S2. Let $d_G(J_l)$ be the number of nets which cross J and which have a terminal in S1, let p_l and p_r be the number of nets selected (as one of the pulled nets) which have a terminal in S1 and S2, respectively, and let x be the number of nets which have a terminal in both S1 and S2.

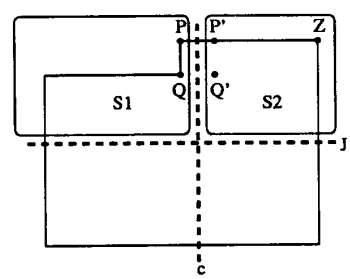

Figure 7. Notation for Subcase 3.2

It is straightforward, but tedious, to figure a lower bound on the total number of terminals to the left of c: $k(2v + h - 1)$; a lower bound on the number of terminals left of c not crossing c: $k(v + j + h - 1) -$

$p_l + (d_G(J_l) - p_l + x)$; and an upper bound on $d_G(c)$: $k(h-1) + \alpha_c + N_c$. From this we obtain $d_G(J_l) \leq -k(h-1-v-j) + 2p_l - \alpha_c + N_c - x$. Performing a similar count on the right side of c yields $d_G(J_r) \leq -k(h-1-n+v-j) + 2p_r - \alpha_c + N_c - x$. Putting the two together, $d_G(J) = d_G(J_l) + d_G(J_r) \leq kn - 2(\alpha_c + N_c - (p_l + p_r - x)) - k(h-j) \leq kn - 2(p - (p_l + p_r - x))$.

It is easy to verify that the density of J increases by no more than $2(p - (p_r + p_l - x))$, so that $d_{G'}(J) \leq kn$. Hence, no horizontal cut becomes over-k-saturated.

Subcase 3.3 $p = \alpha_r - (N_{cr} - a)$). We see that r is not over-k-saturated in G', for $N_r + (N_{cr} - a) + p = N_r + \alpha_r$ nets are routed on PQ, which is the amount necessary. On the other hand, the number of nets routed on PP' is $N_c + a + p \geq N_c + a + \alpha_c - a = N_c + \alpha_c$ which is the amount necessary for c to avoid over-k-saturation in G'. Clearly, no h-cut other than r could be over-k-saturated in G', since all p nets are chosen from those having a terminal on $P'Z$. Only a v-cut $J = (j, j+1)$ to the right of c may become over-k-saturated, and then only if at least one of the p nets chosen by the algorithm has both its terminals to the right of J, for then both of the replacement nets i' and i'' cross J. As in subcase 3.2, a straightforward, but tedious, counting argument will verify that J does not become over-k-saturated. Since the p nets to be pulled are chosen on the basis of their leftmost terminal, one can compute a lower bound on the number of terminals to the right of J belonging to nets which do not cross J. From this, it is possible to show that $d_G(J) \leq km - 2p_{bad}$, where p_{bad} is the number of nets pulled that have both terminals to the right of J. Since the density of J increases by $2p_{bad}$, we have $d_{G'}(J) \leq km$, and J is not over-k-saturated in G'. □

We now show that the last line of the algorithm performs as expected.

Claim. *The possible addition of the fictitious net in the last line of PROCESS(P) causes the remaining k-NRRP to be standard, and does not cause r or c to become over-k-saturated.*

Proof. Suppose k is even. Then, a fictitious net is added only if an odd number of nets are routed on each of PP' and PQ. So, the addition of the fictitious net makes G' standard. To see that r is not over-k-saturated as a result of the new net, note that this could only happen if r were already saturated before the creation of the fictitious net, which would mean $d_{G'}(r)$ is even. But the fictitious net only is added if an odd number of nets are routed on PP', and so only when $d_{G'}(r)$ is odd. A similar argument holds for c.

Now suppose k is odd. Then, a fictitious net is added only if an even number of nets are routed on each of PP' and PQ. Again, the fictitious net makes G' standard. Neither r nor c is over-k-saturated, for this could only happen if r or c were already saturated before the creation of the fictitious net. Again, a parity argument shows this to be impossible. □

Claim. *The time complexity of the algorithm is* $O(knm)$.

Proof. Before the problem is begun, the density of each v-cut and h-cut can be computed, which requires $O(knm)$ time. Following that, the pairing of the gridpoints of odd extended degree requires only $O(n+m)$. The processing of each gridpoint can be performed in $O(k)$ time, and there are nm gridpoints. Hence, the total running time is $O(knm)$. □

This completes the proof of the theorem. □

If no row or column of a k-RRP is k-saturated, then the revised k-ary row and column criteria hold, and we have the following:

Corollary 1 *Given a k-RRP, if there are no k-saturated or over-k-saturated rows or columns, then a layout exists.*

The above results are not known to hold for multiterminal nets, but using the above corollary we can apply the argument used in [MP86] to give an approximation algorithm for the problem.

Theorem 2 *Let R be an* $m \times n$ *multiterminal k-RRP in which no row or column is over-k-saturated. Obtain R′ from R by inserting an empty row (column) between every pair of adjacent rows (columns) of R, except that three empty rows (columns) are inserted between rows (columns) 1 and 2. Then R′, which has size* $(2m+1) \times (2n+1)$, *is routable using k-ary overlap.*

Proof. This follows the proof of [MP86]. View each multiterminal net as a cycle on the boundary of R'. Create a copy of each terminal and move it to an adjacent gridpoint. The cycle can now be viewed as a collection of two-terminal nets. In doing this, we at most double the density of any row or column in R'. Hence, no row or column of R' is k-saturated. By Corollary 1, there is a layout for R'. □

5 Future Work

We have presented an $O(knm)$ time algorithm to construct a layout for the two-terminal net k-RRP, whenever such a layout exists. We believe the approach in [MP86] can be used to reduce the running

time of our algorithm to $O(b \log b)$, where b is the number of gridpoints on the boundary. There are a number of other open issues.

Regions of non-rectangular shape seem likely to be efficiently routable. For example, a *convex grid* is a grid in which every v-cut and h-cut crosses the boundary only twice. From a theorem of [OS81], we know that if every cut of a graph has its capacity greater than or equal to its density then there exists a k-ary overlap layout for it. (The advantage of the algorithm we give is that only v-cuts and h-cuts need to be considered, and so needless computation involving the other cuts is avoided.) Kaufmann and Mehlhorn [KM86] prove that for convex grids if all v-cuts, h-cuts, and 1-bend cuts have density at most equal to capacity, then every cut has its density at most equal to its capacity. This suggests that an efficient algorithm also exists for solvable k-ary overlap routing problems in the convex grid, as again most cuts need not be considered. Under what conditions would a layout exist? Lai and Sprague [LS87] characterize routability in convex grids. In particular, the revised row and column criteria are not sufficient for the existence of a layout for the general problem, but they are sufficient if the problem is standard. We hope to extend [KM86] and [LS87] in order to show that for any k-ary overlap routing problem in a convex grid, there exists a layout iff there is a pairing of the boundary vertices of odd extended degree such that if the pairings are considered to be nets, no v-cut or h-cut is over-k-saturated. Moreover, we expect that such a layout, if one exists, can be constructed in time $O(kN)$, where N is the number of gridpoints in the region.

Our algorithm presented here can be extended to non-rectilinear grids, such as the 60° grid [PBB91]. In this case we do not have v-cuts and h-cuts, but rather three cut directions, running at 60° with respect to each other.

Some other extensions appear more difficult. We gave only a weak multiterminal net result and some improvement may be possible, but the determination of exact conditions for the existence of multiterminal net layouts seems likely to be hard.

Finally, we have not addressed at all the issue of assigning layers to our path edges. For the case of no-overlap (1-RRP's) it is known that four layers suffice [BB84]; i.e., there is an algorithm which wires any no-overlap layout using at most four layers. (This algorithm also works for multiterminal nets and for arbitrary shapes on a rectangular grid.) More recently, Chiang and Sarrafzadeh [Chi91] have shown that 1-RRP's which permit some 2-ary vertical overlap can also be routed using at most four layers. Extensions of this work would be of great interest.

Acknowledgement: We would like to thank Majid Sarrafzadeh for his helpful comments.

References

[BB84] M. Brady and D. J. Brown. VLSI routing: Four layers suffice. In F. P. Preparata, editor, *Advances in Computing Research 2 (VLSI Theory)*, pages 245–257. JAI Press, Inc., Greenwich, CT, 1984.

[BM86] M. Becker and K. Mehlhorn. Algorithms for routing in planar graphs. *Acta Informatica*, 23:163–176, May 1986.

[Chi91] C. Chiang. On wiring overlap layouts. *Proc. First Great Lakes Symposium on VLSI*, pages 25–30, March 1991.

[Fra82] A. Frank. Disjoint paths in a rectangular grid. *Combinatorica*, 2(4):361–371, 1982.

[Fra85] A. Frank. Edge-disjoint paths in planar graphs. *J. of Combinatorial Theory, Series B*, 39:164–178, 1985.

[GK87] S. Gao and M. Kaufmann. Channel routing of multiterminal nets. *Proc. 28th IEEE Symposium on Foundations of Computer Science*, pages 316–325, October 1987.

[KM86] M. Kaufmann and K. Mehlhorn. Routing through a generalized switchbox. *J. of Algorithms*, 7:510–531, 1986.

[LS87] T. Lai and A. Sprague. On the routability of a convex grid. *J. of Algorithms*, 8:372–384, 1987.

[MP86] K. Mehlhorn and F. P. Preparata. Routing through a rectangle. *J. of the Association for Computing Machinery*, 33(1):60–85, 1986.

[OS81] H. Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *J. of Combinatorial Theory*, 31:75–81, 1981.

[PBB91] K. Powers, D. Brown, and M. Brady. The 60° grid: routing channels in width $\frac{d}{\sqrt{3}}$. *Proc. First Great Lakes Symposium on VLSI*, pages 214–219, March 1991.