1984

Optimal Rotation Problems in Channel Routing

Mikhail J. Atallah
Purdue University, mja@cs.purdue.edu

Susanne E. Hambrusch
Purdue University, seh@cs.purdue.edu

Report Number:

84-467

Optimal Rotation Problems in Channel Routing

Mikhail J. Atallah
Susanne E. Hambrusch

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR-467 January 1984

Abstract

In the channel routing problem two rows of terminals, which are opposite each other, have to be connected. We study what effect the rotation of one row of terminals has on the cost measures of the routing phase. The cost measures we consider are the crossing number, which is closely related to the number of crossings between two wires on different layers, the density, which determines the width of the channel, and the length of nets, which is related to the wire length needed in the routing of the channel routing problem. We present efficient algorithms for determining the rotation which minimizes each of these cost measure.

1. Introduction

Conventional design systems for the layout of VLSI chips consist of two interrelated phases: placement and routing. The placement phase arranges terminals, which are connection points for wires, on the chip, and the routing phase determines the necessary interconnections. A common way to cope with the inherently difficult combinatorial problems arising in placement and routing is to partition them into smaller, independent subproblems which can be solved more efficiently [R]. One important subproblem is the channel routing problem (CRP) in which two rows of terminals are positioned opposite each other ([D], [PL], [RBM]). While some placement conditions are determined by technological constraints, placement, in general, contains a certain amount of freedom ([DKSSU], [GCW], [LL], [LeP]). Using this freedom so that it makes routing easier is often a difficult problem.

In this paper we show how the freedom to *rotate* one row of terminals can efficiently be used to minimize some cost measures of the routing phase. Let the two rows of terminals be of length m, and assume we are given n pairs of terminals (p_i, q_i), called *nets*, in which no two nets share a common terminal. We need to connect p_i, which is on the upper row, with q_i, which is on the lower row, $1 \le p_i, q_i \le m$. The lower row of terminals can be circularly rotated before the routing, where a rotation of size ρ results in every q_i moving ρ positions to the right (position 1 is the successor of position m). An example of the effect of a rotation can be seen in Figure 1.1, where (a) shows the initial situation and (b) the situation after a rotation of size 3. The problem we consider is that of choosing a value of rotation which minimizes any of a number of cost measures. Since the cost measures to be minimized are closely related to the wiring model used in the routing phase, we briefly describe the model.

Our wiring model is the rectilinear knock-knee model ([H], [PL], [RBM]) with at least 2 layers. A number of results also hold in polygonal or curvilinear models. The channel (i.e., the space between the two rows) consists of a rectangular grid, and wires have to run along grid lines. Each layer can contain horizontal and vertical wires, and two different wires are allowed to cross or form a knock-knee at a grid-point (as net 2 and net 4 in Figure 1.2), but are not allowed to run on top of each other.

One of our cost measures is the *crossing number* of a CRP. The crossing number of the i-th net (p_i,q_i) is the number of nets which have their entry terminal to the right of p_i while their exit terminal is to the left of q_i, or their entry terminal is to the left of p_i while their exit terminal is to the right of q_i. The crossing number of the CRP is the largest crossing number of any net. For example, nets 1,2,3, and 4 of Figure 1.1(a) have a crossing number of 2,2,3, and 3, respectively, and thus the CRP has a crossing number of 3. Note that in an actual wiring the number of wires that cross the wire joining p_i to q_i has to be at least as large as the crossing number of net i. We give an $O(n \log n)$ time algorithm that determines the rotation which minimizes the crossing number of the resulting CRP. The solution involves an interesting variant of 2-3 trees, one in which an integer key is not stored in any particular node but is "distributed" on a path from the root to a leaf, and is equal to the sum of the data items stored in the nodes of this path. We also outline an $O(n \log n)$ time algorithm for the easier problem of minimizing the sum of the crossing numbers of all nets.

Another cost measure we consider is the *density* of the CRP, which is the maximum over all x of the number of nets (p_i,q_i) for which $p_i \le x < q_i$ or $p_i > x \ge q_i$; i.e., it is the number of nets that have to cross from column x to column $x+1$. For example, the CRP of Figure 1.1(a) has density 3, which is achieved between columns 3 and 4, 4 and 5, and 5 and 6, respectively. Every efficient routing algorithm tries to minimize the width of the channel, and the width is at least as large as the density of the CRP. We show how to determine the rotation achieving minimum density in $O(n^2 \log n)$ time. The third cost measure considered is the *length of nets*, where the length of net i equals the channel width plus the horizontal distance separating p_i and q_i (the length of net i is therefore a lower bound on the length of the wire needed to connect p_i and q_i). Note that the running times of our algorithms depend only on n, not on m (we thus place no bound on how large m can be).

A problem of a similar nature as the rotation problem is the offset problem in which the lower row of terminals can slide right or left. The offset problem with an underlying 1-layer model (i.e., no crossings are allowed) has been studied in [DKSSU]. Our bounds achieved for determining the rotation minimizing the density and the length of nets also hold for the offset problem. In fact, our algorithm for the density becomes simpler when modified to solve the optimal offset problem. In addition, some of our algorithms for minimizing the length of nets are faster for the optimal offset problem than for the optimal rotation problem. In some sense the optimal offset problem is easier than the optimal rotation problem because it does not have the discontinuities introduced by the q_i's jumping from position m to position 1. Note that the crossing numbers are independent of the offset.

The paper is organized as follows. Section 2 contains the algorithms concerning the crossing number. In Section 3 we discuss how to determine the rotation minimizing the density, and Section 4 contains results about the length of nets. Section 5 briefly sketches our results for the optimal offset problem.

2. Minimizing the Crossing Number

In this Section we first consider the problem of finding a rotation which minimizes the crossing number of the resulting CRP, and give an $O(n\log n)$ time algorithm for solving this problem. Since a net with a large crossing number is more vulnerable to electrical interference caused by electrical signals traveling on the other nets, it is important to select a rotation which minimizes the largest crossing number. Furthermore, our algorithm can be used to determine whether there exists a rotation that results in a 1-layer wirable CRP (i.e., a CRP with no crossings).

Throughout this paper we assume that the input consists of the nets (p_i,q_i), $1 \le i \le n$, where $p_1 < p_2 < \cdots < p_n$. Two nets i and j cross each other if $j<i$ and $q_j > q_i$, and the crossing number of a net is the number of nets which cross it. Let $c_i(\rho)$ be the crossing number of net i at rotation ρ. We want to minimize $c(\rho)$, where $c(\rho) = \underset{i}{Max}\, c_i(\rho)$.

Lemma 2.1 The initial crossing numbers $c_1(0), \cdots, c_n(0)$ can be computed in time $O(n\log n)$.

Proof: Let r_i be the rank of q_i among the q_j's at zero rotation. First we compute the r_i's in time $O(n\log n)$, by sorting the q_j's. We now use a modified 2-3 tree structure T whose leaves contain nets ordered according to their r_i values. A leaf containing net i is called *leaf* i. We start with T empty and insert in it the nets $1, 2, \cdots, n$ in that order, updating it in time $O(\log n)$ for each insertion. Every node v of T contains, in addition to the information usually stored in 2-3 trees, (i) $NR(v)$, the number of leaves in v's subtree, and (ii) a number $\Delta(v)$. The significance of the Δ's is that for every leaf i, the sum of the Δ's on the path from the root to i equals the number of intersections between net i and the other nets that are currently stored in T. Once the final tree has been constructed, the $c_i(0)$'s can easily be computed.

The updates required when net i is inserted are guided by the following observation: The number of intersections between net i and the nets already in T is the number of leaves of T that are to the right of leaf i. Every such leaf j has $j<i$ and $q_j>q_i$, and net j is therefore involved in a crossing with net i. Hence we have to record that the sum of the Δ's on the path from the root to every leaf to the right of i is incremented by 1. We do this by adding 1 to the Δ's of the nodes on the right fringe of the path from the root to leaf i. The *right fringe* of a path is the sequence of nodes obtained when going down the path and listing the siblings to the right of the current node (the concept of left fringe is defined in a similar way). More precisely, inserting net i into T is done as follows:

(i) Use r_i to find the location of T where net i will be inserted. This traces a path P from the root to the place of insertion.

(ii) Set $\Delta(i)$ to the sum of the NR's of the nodes on the right fringe of P minus the sum of the Δ's on P. (Note that this ensures that the sum of the Δ's on P plus $\Delta(i)$ equals the number of leaves to the right of i.)

(iii) Add 1 to the Δ of every node on the right fringe of P.

(iv) Insert net i into the 2-3 tree. Maintaining the correct NR and Δ values as nodes get split can easily be done with minor modifications to the standard insertion procedure described in [AHU].

Correctness of the above Steps (i)–(iv) follows from the preceding discussion. The $O(\log n)$ time per insertion follows from the fact that the height of a 2-3 tree (and hence the number of nodes on P and on its fringes) is $O(\log n)$. Since there are n insertions, their total cost is $O(n \log n)$. After the n nets have been inserted, $c_1(0), \cdots, c_n(0)$ can computed in time $O(n)$ by a preorder traversal of the tree T. ∎

Let $\rho_1 \leq \cdots \leq \rho_n$ be the rotations at which the q_j's have changed from position m (at $\rho_i - 1$) to position 1 (at ρ_i). For example, in Figure 1.1, we have $\rho_1=2$, $\rho_2=3$, $\rho_3=5$, and $\rho_4=6$. We want to select the ρ_i which minimizes c.

Theorem 2.2 It is possible to compute, in time $O(n\log n)$, the minimum crossing number and a value of rotation which achieves it.

Proof: It is sufficient to show that $c(\rho_1), \cdots, c(\rho_n)$ can be computed in $O(n\log n)$ time. First we compute $c_1(0), \cdots, c_n(0)$ in time $O(n\log n)$ (Lemma 2.1). At rotation ρ, the tree $\hat{T}(\rho)$ is defined as follows. $\hat{T}(\rho)$ is a modified 2-3 tree whose leaves contain the n nets. Net i is stored in the i^{th} leftmost leaf of $\hat{T}(\rho)$, which is referred to as *leaf* i. (Note that $\hat{T}(\rho)$ differs substantially from T of Lemma 2.1's proof). Every node v of $\hat{T}(\rho)$ has two additional fields:

(i) A $\Delta(v)$ field, whose significance is that for every leaf i, $c_i(\rho)$ equals the sum of the Δ's of the nodes on the path from the root to i.

(ii) A $MAX(v)$ field, which contains $\Delta(v)$ plus the largest of the MAX's of v's children.

Initially (i.e., at rotation zero), leaf i of $\hat{T}(0)$ has $\Delta(i)=c_i(0)$ $(1 \leq i \leq n)$, and the Δ of every internal node is zero. Given the $c_i(0)$'s, building $\hat{T}(0)$ and computing $MAX(v)$ for every node v can be done in $O(n)$ time.

The MAX value at the root of $\hat{T}(\rho)$ is equal to $c(\rho)$. This can be seen by observing that for every v, the definition of $MAX(v)$ implies that it is the maximum, over all leaves i in the subtree of v, of the sum of the Δ's on the v-to-i path. Hence, MAX at the root is the largest of $c_1(\rho), \cdots, c_n(\rho)$, which is $c(\rho)$.

Starting with $\hat{T}(0)$, we compute the sequence $\hat{T}(\rho_1), \hat{T}(\rho_2), \cdots \hat{T}(\rho_n)$. Every time we change $\hat{T}(\rho_{i-1})$ into $\hat{T}(\rho_i)$, we examine the MAX value at the root (which is equal to $c(\rho_i)$), and at the end we select the rotation ρ_i for which the $c(\rho_i)$ is smallest. In order to complete the proof, it suffices to show how to obtain $\hat{T}(\rho_i)$ from $\hat{T}(\rho_{i-1})$ in $O(\log n)$ time. Assume ρ_i causes q_j to move from position m to 1. The effect on the c_k's caused by q_j moving from position m to position 1 is that net j switches from crossing nets $j+1, \cdots, n$ (Figure 2.1(a)) to crossing nets $1, \cdots, j-1$ (Figure 2.1(b)). The update should therefore add $2j-n-1$ to c_j, increment (resp. decrement) by one c_k for every leaf k which is to the left (resp. right) of j, and update the appropriate *MAX* values. This is implemented in time $O(\log n)$ as follows:

(i) Trace a path P from the root to leaf j and add $2j-n-1$ to $\Delta(j)$

(ii) add 1 to the Δ's of the nodes on the left fringe of P, and add -1 to the Δ's of the nodes on the right fringe of P

(iii) update the *MAX* value of the nodes on path P and its left and right fringes (these are the only nodes whose *MAX* needs updating)

Hence, $T(\rho_i)$ can be obtained from $T(\rho_{i-1})$ in time $O(\log n)$, which concludes the proof of Theorem 2.2. ▪

The ***total crossing number*** of a CRP is half the sum of the crossing numbers of all nets. The algorithm for determining the rotation ρ that minimizes the total crossing number is similar to the one outlined above. The initial total crossing number is computed in $O(n \log n)$ time (Lemma 2.1), and the optimal ρ can be determined in $O(n)$ time. Since we do not need to keep track of the number of nets crossing each individual net, only the total number, the algorithm requires less bookkeeping.

We now briefly sketch a proof that the $O(n \log n)$ bound for the total crossing number is optimal. We do this by reducing to this problem, in $O(n)$ time, the problem of computing the total number of inversions in a permutation. In a comparison-based model, $O(n \log n)$ comparisons are needed for computing the total number of inversions in an n-element permutation [K].

Let $a_1\, a_2 \cdots a_n$ be a permutation of $\{1,2, \cdots n\}$. Let the CRP consisting of the nets $(a_i, n+i)$, $1 \le i \le n$, be the *basic* CRP. It is easy to see that the total crossing number in the basic CRP is equal to the total number of inversions in the given permutation. Next embed the basic CRP in an $4n$-net CRP as shown in Figure 2.2. The total crossing number of the new CRP (at rotation zero) is $2n^2+2c$, where c is the total crossing number in the basic CRP. No rotation can achieve a total crossing number less than $2n^2+2c$. Hence, the total number of inversions can be obtained by solving the crossing number problem.

3. Minimizing the Density

We give an $O(n^2 \log n)$ time algorithm for computing the rotation which minimizes the density. Minimizing the density d will also minimize the channel width required in the routing phase. When 2 layers are available, $2d-1$ is both an upper ([RBM]) and a lower ([L]) bound on the channel width. When 3 or more layers are available, the channel width equals the density ([PL]).

The algorithm keeps track of the density as the rotation increases from zero to $m-1$ and remembers which rotation achieves lowest density. We use $dp(i,\rho)$ (resp. $dq(i,\rho)$) to denote the "local" density just to the right of terminal p_i (resp. q_i) at rotation ρ. For example, in Figure 1.1(a) the dp's are 1,3,1,0 and the dq's are 3,1,3,2. In addition, for $0 \le i \le n$, we define $c(i,\rho)$ as the number of $dp(j,\rho)$'s and $dq(j,\rho)$'s whose value is i. More formally:

$$c(i,\rho) \;=\; |\{j \,|\, dp(j,\rho)=i\}| \;+\; |\{j \,|\, dq(j,\rho)=i\}|.$$

For example, in Figure 1.1(a) the c's are 1,3,1,3,0. The density at rotation ρ, $d(\rho)$, is the largest of the $dp(i,\rho)$'s and $dq(i,\rho)$'s, and it can equivalently be thought of as the largest j for which $c(j,\rho)$ is nonzero, i.e.:

$$d(\rho) = \underset{1 \le i \le n}{Max}\, Max\{dp(i,\rho), dq(i,\rho)\} = Max\{j \,|\, c(j,\rho) > 0\}.$$

When the value of the rotation is clear from the context, we no longer explicitly include the dependence on ρ in the above functions (e.g., we say $dp(i)$ rather than $dp(i,\rho)$).

Let ϑ_{ij} be the rotation in which p_i and q_j coincide. For example, in Figure 1.1 we have $\vartheta_{11}=3$, $\vartheta_{12}=2$, and $\vartheta_{23}=2$. As in Section 2, $\rho_1 \le \cdots \le \rho_n$ are the rotations at which the q_j's change from position m (at $\rho_i - 1$) to position 1 (at ρ_i). The density changes at no more than $2n^2+n$ distinguished values of the rotation, namely as the rotation increases (i) from some $\rho_i - 1$ to ρ_i, or (ii) from some $\vartheta_{ij} - 1$ to ϑ_{ij}, or (iii) from some ϑ_{ij} to $\vartheta_{ij}+1$. We assume that those $2n^2+n$ distinguished values of the rotation are all distinct (this is assumed to make the exposition simpler, and the algorithm can easily be modified to handle the general case).

We now examine the above cases more carefully.

Case (i) corresponds to q_j moving from position m (Figure 2.2(a)) to position 1 (Figure 2.2(b)). This motion decreases by 1 each $dp(k)$ with $p_k \geq p_j$ and each $dq(k)$ with $q_k \geq p_j$, and it increases by 1 each $dp(k)$ with $p_k < p_j$ and each $dq(k)$ with $q_k < p_j$.

Case (ii) corresponds to q_j moving from being one unit to the left of p_i (Figure 3.1(a)) to being in the same column as p_i (Figure 3.1(b)). The only change due to this motion is in $dq(j)$: It increases by 1 if $q_i > q_j$ (as in Figure 3.1), and decreases by 1 if $q_i \leq q_j$.

Case (iii) corresponds to q_j moving from being in the same column as p_i (Figure 3.2(a)) to being one unit to the right of p_i (Figure 3.2(b)). The only change due to this motion is in $dp(i)$: It increases by 1 if $p_i \geq p_j$ (as in Figure 3.2), and it decreases by 1 if $p_i < p_j$.

Whenever we change $dp(i)$ or $dq(i)$ we also update the appropriate $c(j)$'s and d, the current density. If $dp(i)$ or $dq(i)$ increases from k to $k+1$ then we must decrement $c(k)$ by one, increment $c(k+1)$ by one, and if d was equal to k we set it equal to $k+1$. If $dp(i)$ or $dq(i)$ decreases from k to $k-1$ then we increment $c(k-1)$ by one, decrement $c(k)$ by one, and if this causes $c(k)$ to become zero and d was equal to k then we set d equal to $k-1$.

The following is an informal outline of the algorithm.

1.) Compute, at zero rotation, all the $dp(i)$'s, $dq(i)$'s, $c(i)$'s, and d. This can be done in time $O(n \log n)$.

2.) Compute the ϑ_{ij}'s and the ρ_i's, and then sort, in time $O(n^2 \log n)$, the sequence of $2n^2+n$ values of rotations corresponding to the ϑ_{ij}'s, $(\vartheta_{ij}-1)$'s, and (ρ_i-1)'s.

3.) Scan the sorted sequence obtained in the previous step. For each entry, update the appropriate dp's and/or dq's, in the manner outlined above. There are $2n^2$ occurrences of Cases (*ii*) and (*iii*), each of which requires $O(1)$ time for updating. There are n occurrences of Case (*i*), each of which requires $O(n)$ time for updating. Therefore this Step takes $O(n^2)$ time.

The above algorithm produces $2n^2+n$ values of d, one for each of the distinguished values of the rotation. Therefore it is possible to find the smallest possible density together with a rotation which achieves it in time $O(n^2 \log n)$.

4. Minimizing the Net Length

In this Section our cost measure is the *length* of the nets. Recall that the length of net i equals the channel width (i.e. the distance between entry and exit track) plus the horizontal distance separating p_i and q_i. For example, in Figure 1.2 the length of net 2 is 2+3=5. In an actual wiring the wire length needed to connect p_i and q_i is at least as large as the length of net i. In some situations the channel width will already be fixed and only the rotation can be varied. This causes the vertical portions of the lengths of all nets to be the same, and we need only consider the horizontal portion. Hence, we distinguish the following 3 cases.

Case (i) The channel width is fixed and large enough to wire the CRP generated by every rotation.

Case (ii) The channel width is fixed, and the only CRP's to be considered are those with a density not exceeding a given value s.

Case (iii) The channel width can be varied and thus the vertical portions of the lengths of nets must be taken into account.

For each of Cases (i) to (iii), we consider the two problems of determining a rotation that minimizes the total length of all nets, and determining a rotation that minimizes the maximum length of any net. For Case (i), these two problems can be solved in $O(n \log n)$ and $O(n^2)$ time, respectively, and for Cases (ii) and (iii) both problems can be solved in $O(n^2 \log n)$ time.

We first describe the solutions for the problem of minimizing the *total net length* (i.e., the sum of the lengths of all nets). For Case (i), the vertical portions of the lengths of all nets are the same and are fixed. Therefore we need only consider the horizontal portions. Let $l_i(\rho)$ be the horizontal portion of the length of net i at rotation ρ. How l_i varies as a function of ρ can be described by a list of length $O(1)$, and therefore $L(\rho)=\sum_i l_i(\rho)$ can be described by a list of length $O(n)$. The list describing L can be obtained in time $O(n \log n)$ as follows:

Recursively compute the list describing $\sum_{i=1}^{n/2} L_i$ and the list describing $\sum_{i=n/2+1}^{n} L_i$, then from these two lists obtain in time $O(n)$ that describing L. (The last step is done in a manner reminiscent of the way two sorted sequences are merged, and we leave its details to the reader.) Once we have the list describing L, we can determine in $O(n)$ time the minimum value of L as well as a rotation which minimizes it.

We now briefly describe how to obtain the optimal rotation minimizing the total net length in Case (ii), when a maximum allowable density s is given. As in Case (i), we need only consider the horizontal portions of the lengths of nets, but now we must ignore rotations which result in a density exceeding s. Observe that we need only consider the rotations at position $\vartheta_{ij}-1$, ϑ_{ij}, and ρ_i-1 (these quantities are defined as in Section 3). Use the algorithm given in Section 3 to obtain the sorted sequence of $2n^2+n$ values of density corresponding to the $2n^2+n$ distinguished values of rotation. Scan the sequence from left to right and determine for each rotation b_k the total horizontal net length nl_k, $1 \le k \le 2n^2+n$. Let d_k be the density corresponding to rotation b_k. After an initial computation of nl_1 in time $O(n)$, we can obtain nl_k from nl_{k-1} in constant time. From the nl_k's obtained, we discard those for which $d_k > s$. >From the surviving nl_k's, we choose the smallest. Hence the problem can be solved in $O(n^2 \log n)$ time.

The optimal rotation that minimizes the total net length when the channel width is not fixed can also be obtained in time $O(n^3 \log n)$, in a manner similar to the one outlined above (we omit the details).

We now turn our attention to the problem of finding a rotation which minimizes the *maximum length of any net*. For Case (i), we want to minimize $l(\rho)$, where $l(\rho) = \underset{1 \le i \le n}{Max}\, l_i(\rho)$. We show that a rotation which minimizes l can be computed in time $O(n^2)$. First, we note that the list describing how l varies as a function of ρ has a length of $O(n^2)$. This means that, if we recursively compute the list describing $\underset{1 \le i \le n/2}{Max} l_i$ and the list describing $\underset{n/2+1 \le i \le n}{Max} l_i$, then we can obtain from these two lists the list describing l in time $O(n^2)$. Therefore, if we let $T(n)$ denote the worst-case time needed to compute the list describing l, then we have $T(n)=2T(n/2)+cn^2$, which implies that $T(n)=O(n^2)$. Once we have the list describing l, we scan this list and get the minimum value of l as well as a value of rotation which achieves it. (We conjecture that the length of the list describing l is actually $O(n)$ rather than $O(n^2)$, which would imply that $T(n)=O(n \log n)$.)

The problem of minimizing the largest length of any net for Cases (ii) and (iii) can be solved in time $O(n^2 \log n)$, in a manner similar to that used for minimizing total net length.

5. Optimal Offset Problems

In this Section we briefly sketch algorithms for choosing the offset which minimizes the density, the total net length, and the length of the longest net, respectively. Throughout this Section, we assume that zero offset corresponds to a situation where all the q_i's are to the left of p_1.

The problem of choosing the offset which minimizes the *density* can be solved in time $O(n^2 \log n)$ by an algorithm similar to the one given in Section 3 for the rotation problem. The details are omitted, but it is worth noting that the algorithm for the offset problem is simpler than the one for the rotation problem. We no longer have exit terminals jumping from position m to position 1. The same result for minimizing the density in the offset problem has been reported in [LaP].

We next consider the problem of finding the offset which minimizes the *total net length*. We again distinguish between the 3 cases described in Section 4. In Case (i), when the channel width is fixed and large enough to wire any CRP, the optimal offset can be found in time $O(n)$, as follows. Let $l_i(\vartheta)$ be the length of the horizontal portion of net i at offset ϑ, and let ϑ_i be the offset at which net i is trivial (i.e. $l_i(\vartheta_i)=0$). Note that $l_i(\vartheta) = |\vartheta - \vartheta_i|$. Let $L(\vartheta) = \sum_i l_i(\vartheta)$, and observe that L is continuous and is piecewise linear. Since L is the sum of convex functions, it is also convex. An offset which minimizes L can be computed in time $O(n)$ by noting that L has a local minimum at the median of the ϑ_i's. Since L is convex this is also a global minimum. Finding the ϑ_i's (not sorted) can be done in time $O(n)$, after which finding their median also takes time $O(n)$ [BFPRT]. For Cases (ii) and (iii), an argument similar to that given in Section 4 shows that the time bounds are $O(n^2 \log n)$.

Minimizing the *length l of the longest net* in Case (i), can now be done in

$O(n)$ time (compared to $O(n \log n)$ for the rotation problem). Recall that $l(\vartheta) = \underset{1 \le i \le n}{Max}\, l_i(\vartheta)$. Find i (resp. j) such that $l_i(0)$ (resp. $l_j(0)$) is smallest (resp. largest), and compute the value of ϑ for which $l_i(\vartheta)=l_j(\vartheta)$. Note that the value of ϑ thus computed minimizes l. For Cases (ii) and (iii), an argument similar to that given in Section 4 shows that the time bounds are $O(n^2 \log n)$.

References

[AHU] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[BFPRT] M. Blum, R.M. Floyd, V.R. Pratt, R.L. Rivest, R.E. Tarjan, 'Time Bounds on Selection', *J. Computer and System Sciences*, 7:4, pp 448-461, 1972.

[D] D.N. Deutsch, 'A Dogleg Channel Router', *Proceedings of the 13th IEEE Design Automation Conf.*, pp 425-433, 1976.

[DKSSU] D. Dolev, K. Karplus, A. Siegel, A. Strong, J.D. Ullman, 'Optimal Wiring between Rectangles', *Proceedings of the 13th Annual ACM Symp. on Theory of Computing*, pp 312-317, 1981.

[GCW] I.S. Gopal, D. Coppersmith, C.K. Wong, 'Optimal Wiring of Movable Terminals', *IEEE Trans. on Computers*, Vol. c-32, 9, pp 845-858, 1983.

[H] S.E. Hambrusch, 'Using Overlap and Minimizing Contact Points in Channel Routing', *Proc. of 21-st Allerton Conf. on Commun., Control, and Comp.*, pp 256-258, 1983.

[K] D.E. Knuth, *The Art of Computer Programmin, Vol. III: Sorting and Searching*, Addison-Wesley, 1973.

[L] F.T. Leighton, 'New Lower Bounds for Channel Routing', unpublished manuscript, 1981.

[LL] H.W. Leong, C.L. Liu, 'A New Channel Routing Problem', *Proc. of 20-th Design Automation Conference*, 1983.

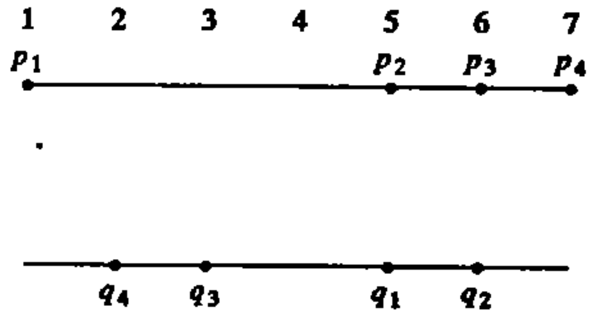
[LaP] A.S. LaPaugh, R.Y. Pinter, 'On Minimizing Channel Density by Lateral Shifting', *Proc. of IEEE Internat. Conf. on Computer Design*, 1983.

[LeP] C.E. Leiserson, R.Y. Pinter, 'Optimal Placement for River Routing', *SIAM*, Vol. 12, Nr. 3, pp 447-462, 1983.

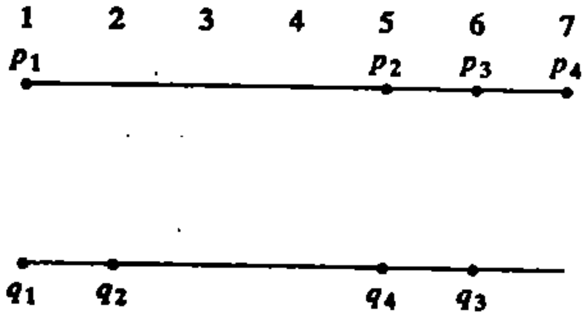
[PL] F.P. Preparata, W. Lipski, 'Three Layers are enough', *Proceedings of the 23rd Annual IEEE Foundations of Comp. Sc. Conf.*, pp 350-357, 1982.

[R] R.L. Rivest, 'The PI (Placement and Interconnect) System', *Proc. of 19-th Design Automation Conference*, pp 475-481, 1982.

[RBM] R.L. Rivest, A.E. Baratz, G. Miller, 'Provably Good Channel Routing Algorithms', *Proceedings of the CMU Conf. on VLSI Systems and Computations*, pp 153-159, 1981.



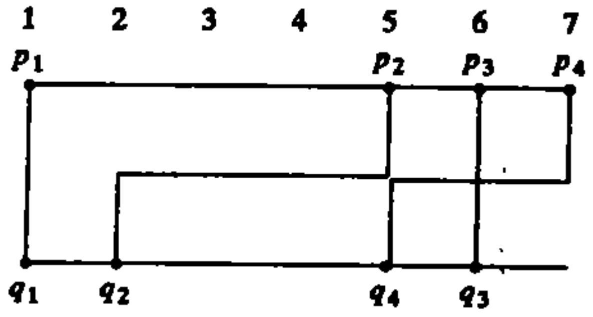
CRP in initial configuration
(zero rotation)
(*a*)



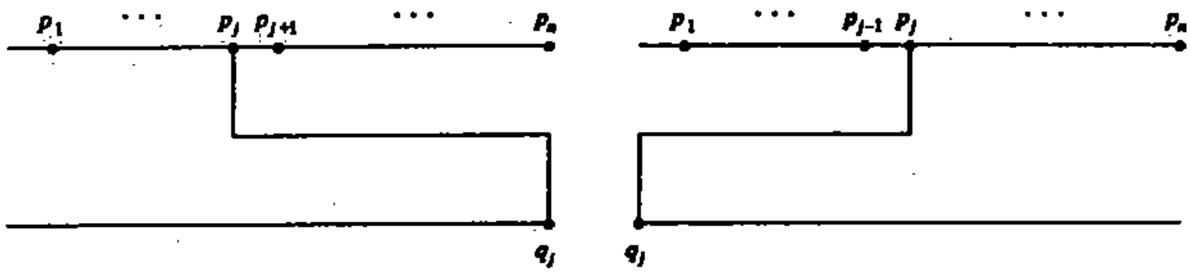
CRP at rotation 3

(*b*)

Figure 1.1



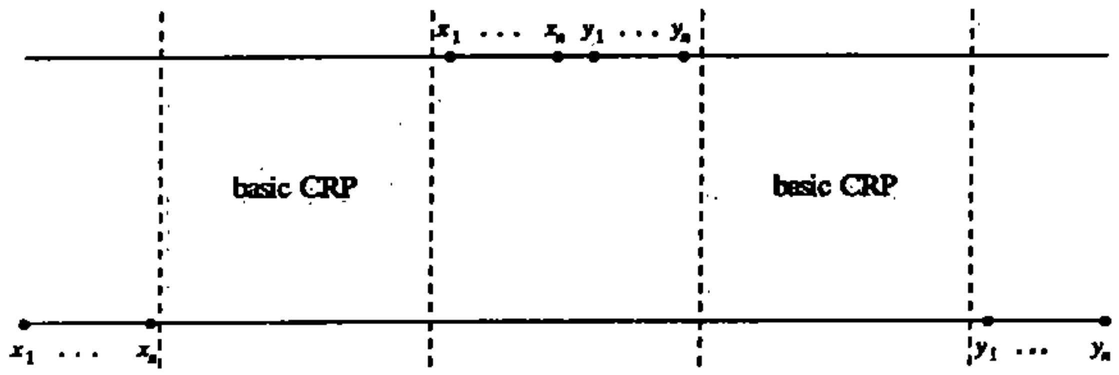
A wiring of the CRP of 1.1(*b*)
Figure 1.2



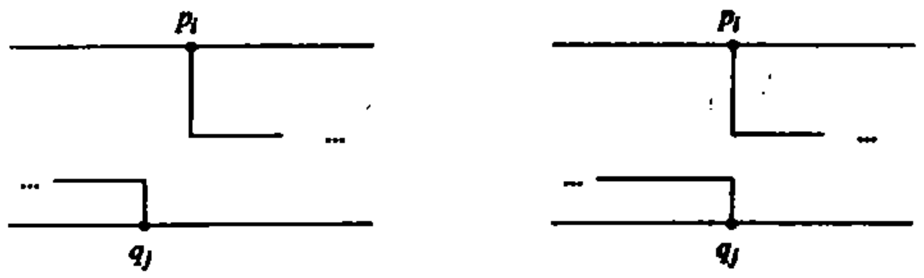
crossing nets j+1, ..., n
(a)

crossing nets 1, ..., j-1
(b)

Figure 2.1



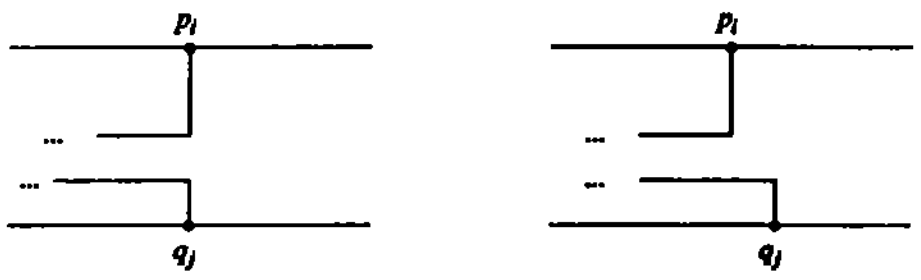
minimum total crossing number achieved at rotation 0 and 4n
Figure 2.2



case (*i*) when $q_i>q_j$

(*a*) (*b*)

Figure 3.1



case (*ii*) when $p_i>p_j$

(*a*) (*b*)

Figure 3.2