A HEURISTIC FOR MANHATTAN ROUTING

E. Lodi
F. P. Preparata

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.
# A Heuristic for Manhattan Routing

**Author(s):** Lodi, E and Preparata, F.P.

**Type of Report:** Technical, final

**Date of Report:** May 1986

**Page Count:** 14

**Subject Terms:** Manhattan routing, channel routing, VLSI

In designing the layout of VLSI chips, channel routing plays a central role. The traditional model for channel routing is known as Manhattan routing, where a two-layer channel is used to route a specified set of nets between two rows of terminals. If the net list contains long runs, the only way to reduce the number of channel tracks is by splitting horizontal segments of nets (jogging).

A new algorithm is proposed which uses preprocessing to decide, before producing the routing, which nets must be split and where. This is achieved by finding a parameter $m$ which represents the maximum length desirable for a run and minimizing the number of runs with length greater than $m$. The minimization is obtained by finding a minimum-cost max-flow in a directed graph $G=\langle V,E \rangle$. Node set $V$ is partitioned in five sets: $\{s\}$, $\{t\}$, $J,A$, and $B$, where $s$ is the source; $t$ is the sink; $N$ is a set of jogging request; nodes in $A$ are associated to nets chosen as possible candidates for joggings; nodes in $B$ are associated to channel columns. All arcs have capacities equal to one and their costs are given.
accordingly to certain priorities assigned to the vertices of A and B. The algorithm is coded in Pascal and implemented on a VAX 11/780 computer and its running time is \( O(n^2) \), where \( n \) denotes the number of nets. Experimental results are particularly satisfactory when runs of quite different lengths can be reduced to the same length.
A HEURISTIC FOR MANHATTAN ROUTING*

E. Lodi** and F.P. Preparata
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

Abstract

In designing the layout of VLSI chips, channel routing plays a central role. The traditional model for channel routing is known as Manhattan routing, where a two-layer channel is used to route a specified set of nets between two rows of terminals. If the net list contains long runs, the only way to reduce the number of channel tracks is by splitting horizontal segments of nets (jogging).

A new algorithm is proposed which uses preprocessing to decide, before producing the routing, which nets must be split and where. This is achieved by finding a parameter m which represents the maximum length desirable for a run and minimizing the number of runs with length greater than m. This minimization is obtained by finding a minimum-cost max-flow in a directed graph \( G = (V, E) \). Node set \( V \) is partitioned in five sets: \( \{s\} \), \( \{t\} \), \( J \), \( A \), and \( B \), where \( s \) is the source; \( t \) is the sink; \( J \) is a set of jogging request; nodes in \( A \) are associated to nets chosen as possible candidates for joggings; nodes in \( B \) are associated to channel columns. All arcs have capacities equal to one and their costs are given accordingly to certain priorities assigned to the vertices of \( A \) and \( B \). The algorithm is coded in Pascal and implemented on a VAX 11/780 computer and its running time is \( O(n^2) \), where \( n \) denotes the number of nets. Experimental results are particularly satisfactory when runs of quite different lengths can be reduced to the same length.

*This work has been supported by the Semiconductor Research Corporation under Contract 83-1 035.

**On Leave from the Dipartimento di Scienze dell'Informazione, University of Pisa, Italy.
1. Introduction

Several models have been used in the study of channel routing. The traditional model known as Manhattan routing assumes a two-layer grid, in which metal wires run on the top grid and poly wires on the bottom. One layer is used only for vertical segments, the other one for horizontal segments; a contact cut (via) is introduced for each layer change of a wire. Szymansky [1] showed that Manhattan routing is NP-hard.

Although introduced in the earlier seventies [2], channel routing has been the subject of several heuristics in the past [3,4,5,6,7,8] and in recent years [9,10,11,12,16]. In fact, channel routing is a key problem in the development of automated layout systems for integrated circuits. Moreover, the whole interconnecting phase can be viewed as a collection of channel routing problems.

After dividing the entire layout region into nonoverlapping rectangular "channels", a global router determines which nets pass through each channel, and then a local channel router computes the detailed routing within each channel.

To address more precisely the problem, we can define a channel as follows: a channel of width $t$ consists of a unit grid, with grid points $(x,y)$ such that $x$ and $y$ are integers in the range $[1,c]$ and $[0,t+1]$ respectively. The vertical lines are called columns and the horizontal lines are called tracks. A two-terminal net $N_i$ is an ordered pair of integers $(p_i,q_i)$ where $p_i$ is the entry column of the net and $q_i$ is the exit column. $N_i$ specifies that the point $(p_i,0)$ must be connected by a wire to the point $(q_i,t+1)$; these points are called entry and exit terminals respectively. The connection between the two terminals is made by a wire using a path of wire segments laid out on the unit grid. A net $N=(p,q)$ is trivial if $p=q$. A column is empty if it has no terminal.

A channel routing problem is a set of nets $S=\{N_1, \ldots, N_n\}$ where no two nets share an entry or exit column. (See Fig.1 for an example of a two-terminal-net channel routing problem, where nets are represented as arrows; the direction of the arrows goes from the entry terminal to the exit terminal.)
A solution to a channel routing problem is an integer $t$, specifying the channel width, and a set of wires satisfying the restriction that two distinct wires can meet at a grid point only if it is a crossover point (see Fig. 2). The major objective of the problem is to minimize the number of tracks used to realize the routing. In addition to the width of the channel, a solution should also aim at minimizing other features, such as the number of additional columns, the maximum wire length, the number of vias, and the total wire length.

An important parameter of a channel routing problem is its density $d$, defined as the maximum over all $x$, of the number of nets $N_i = (q_i, p_i)$ crossing $x$, i.e., the nets for which either $p_i \leq q_i$ or $q_i \leq x < p_i$ (in Fig. 1, $d = 6$). Obviously, the density is a lower bound on the minimum achievable channel width. It has been shown that a slightly different layout mode (the knock-knee mode, where two distinct wires can change direction at the same grid point) allows one to come...
within a factor of two of channel density with two layers [13] and to achieve density with three layers [14]. Unfortunately, for the Manhattan mode, the density alone does not give, in general, a meaningful measure of the routing requirements. In [15] it has been proven that at least $\sqrt{2n}$ tracks are necessary for any problem which has $n$ non-trivial nets spanning $n+1$ contiguous columns. In [16] a similar result was obtained, valid also for multi-terminal nets, introducing a new measure called flux. Roughly speaking, the density measures the number of nets split by a vertical cut of the channel, while the flux measures the number of nets split by a horizontal cut of the channel. More precisely, a routing problem has flux $f$ if $f$ is the largest integer for which there exists a horizontal cut of the channel spanning $2f^2$ columns and splitting at least $2f^2 - f$ nontrivial nets (in Fig. 1 $f = 3$). In [16] an algorithm is also given that produces a Manhattan routing with $d + O(f)$ tracks for every two-terminal-net problem with density $d$ and flux $f$. The major feature of this algorithm consists of supplying a routing with width at most a constant times the optimal value for all problems, although its behavior is not as good as other heuristics for most practical problems.

2. Preliminaries

The model used is such that the wires of any two nets $N_i$ and $N_j$ cannot overlap in a column. If a column is the exit and entry column of $N_i$ and $N_j$ respectively and if each net is allowed just one horizontal segment, the horizontal segment of $N_i$ must be necessarily placed in a track above the one used for $N_j$. These vertical constraints can be represented by a directed graph $G_v$, where each node corresponds to a net and an arc directed from $N_i$ to $N_j$ means that the horizontal segment of $N_i$ must be placed above that of $N_j$. Each connected component of $G_v$ is called a run, and the number of its nodes is called the length of the run. If $N_i$ and $N_j$ are two nets of a run, $d(N_i, N_j)$ is called their distance and is equal to the number of arcs in the directed path between $N_i$ and $N_j$. For the problem of Fig. 1, the vertical constraint graph is shown in Fig. 3 (a) where there are six connected components corresponding to six runs whose lengths are 11, 5, 2, 1, 1 and 3, respectively.
We define *jogging* or *doglegging*, the operation of splitting the horizontal segment of a net into two or more subsegments and of connecting two consecutive subsegments by means of a vertical segment. It is clear that without jogging some nets, the longest run length is a lower bound to the channel width; in addition, jogging is necessary to handle cycles (see Fig. 4). In Fig. 4 we show a jogging using an empty column; however, splitting of a horizontal segment adds a new node to $G_v$ and changes the vertical constraints, resulting in a restructuring of $G_v$ as we shall now discuss.

Fig. 5 shows the rearrangement of a run $r_1$ whose net $N_i$ is split at an empty column; in Fig. 5a we show the layout features and in Fig. 5b we show the modification of the portion of $G_v$ pertaining to the run in question.
Problem specification

layout

empty column

Fig. 4. A jogging used to solve a cycle

Fig. 5. Jogging at an empty column: run $r_1$ is effectively split into two shorter runs $\alpha_1 N_1'$ and $N_1'' \beta_1$.

Fig. 6 shows the rearrangement of a run $r_1$ split at a partially occupied column, which is the exit column of a net $e$ belonging to another run $r_2$. (Notice that $e$ is the rightmost net of run $r_2$.) The net effect is that $r_1$ is split into two runs $\alpha_1 N_1'$ and $N_1'' \beta_1$, the latter of which is then spliced with $r_2$, of which it becomes the termination. We say in this case that $r_1$ and $r_2$ are mated at the jogging column. The case in which the jogging column is also partially occupied, but is the entry column of a net $h$ of a run $r_2$, is illustrated in Fig. 7 and deserves no additional comment.

Finally we consider the most general case where net $N_i$ of run $r_1$ is jogged at a fully occupied column; this column is necessarily the entry and exit column, respectively, for two consecutive nets of a run $r_2$. The net result of the jogging is that the two runs are both split and cross-spliced (i.e., the initial segments of one with the final segments of the other, and vice versa). The situation is
Fig. 6. Jogging at a partially occupied column (exit column): run $r_1$ is split into two runs, one of which is then spliced (mated) with run $r_2$.

Fig. 7. Jogging at a partially occupied column (entry column): run $r_1$ is split into two runs, one of which is then spliced with run $r_2$.

As an additional example, Fig. 3b shows how the graph of Fig. 3a must be changed if the net (8,11) of Fig. 1 is split at column 9. Obviously nets of different runs can be put on the same track if they do not overlap horizontally. As mentioned earlier, to route a given net problem with density $d$ and flux $f$ the minimum channel width is at least $\max(d, Cf)$, for some constant $C$, but it
is not entirely clear how these two measures interact. Clearly the density is an achievable bound on
the number of tracks in problems consisting of runs of length one. On the other hand, consider the
simple channel routing problem whose \( n \) nets represent each a shift to the right by one position. In
this case, \( d \) is equal to one and \( f \) is equal to \( \sqrt{n} \): the flux represents quite well the problem
difficulty [15]. The algorithm proposed in [16] distributes empty columns uniformly across the
channel, thereby dividing the channel columns into blocks of size \( f \), each containing three empty
columns. Therefore it makes use of a number of joggings proportional to \( f \). Intuition suggests
that this approach is successful when the flux can be related to the average length of runs, that is,
the \( f \) empty columns will represent the first entry columns and last exit columns of \( f \) runs each
of which has length \( 2f^{-1} \). Following this idea, a new heuristic is proposed that uses a parameter \( m \)
to classify the runs into three sets \( I_r \), \( S \), and \( L_r \). The nets belonging to runs of \( I_r \) and \( S \) will not
be split, while several nets belonging to runs of \( L_r \) will be split at an empty column or at a column
occupied by net terminals of the runs in \( S \).

3. Description of the Algorithm

The algorithm consists of four phases: the first three phases are preparatory for the routing
that takes place in the fourth phase.

In the first phase runs are listed and the flux \( f \) of the problem is computed. The parameter
\( m \) is selected as the maximum desirable run length. Normally \( m \) is set equal to \( 2f \); however.
because there are problems in which flux either underestimates or overestimates the attainable average length of runs, \( m \) can be also defined by the user, independently of the flux. As a result, all runs are partitioned into three sets \( L \), \( S \), and \( I \). \( L \) contains the runs of length greater than \( m \) (long runs), \( S \) contains the runs of length less than or equal to \( m/2 \) (short runs) and \( I \) contains all the remaining runs (intermediate runs). The runs in \( L \) and in \( S \) are processed in the second and third phase respectively, while the runs in \( I \) are processed in the last phase. For the problem of Fig. 1, choosing \( m = 4 \), we have \( L = \{r_1, r_2, r_3, r_4, r_5\} \) and \( I = \{r_6\} \).

In the second phase each run in \( L \) is scanned to find the target nets, that is, the nets selected for splitting. (Once a net has been selected for splitting, it is subsequently necessary to select a column, within the net span, where jogging must take place; this will be done in the third phase.)

In a given run we require that the distance \( d(N_i, N_j) \) between two consecutive nets \( N_i \) and \( N_j \) be equal to \( m - 1 \). Note, however, that \( d(N_i, N_j) \) may be slightly different from \( m - 1 \), if this is helpful in reducing the total horizontal overlap among target nets and the increase of density that joggings can produce. In fact the jogging of the horizontal segment of a net can increase the density by one or two units above the density of the original problem (see Fig. 9). Furthermore, suppose that in Fig. 9 \( N_1 \) corresponds to the target node in \( G \); comparing the situations illustrated in (a) and (b), it may turn out that the selection of \( N_2 \) as target net may be more advantageous. For this reason, for each target net \( N_i \) a few nets adjacent to \( N_i \), whose collection is denoted as \( \text{NEIGHBOR}(N_i) \) are considered as of possible candidates for jogging. The nets in \( \text{NEIGHBOR}(N_i) \) have associated priorities (as indices of desirability) which are equal to their distance from \( N_i \) in \( G \). Let \( T \) be the set of possible candidates for jogging, each with an associated priority.

For the problem of Fig. 1, the target nets of \( r_1 \) are the nets \( d = (2, 8), g = (15, 18) \) and \( j = (27, 21) \); the target net of \( r_2 \) is \( q = (10, 7) \). All the \( \text{NEIGHBOR}(\ ) \) sets of these nets have cardinality equal to one and are \( c = (5, 2), f = (11, 15), i = (29, 27) \) and \( p = (14, 10) \) respectively. Thus, \( T = \{c, d, f, g, i, j, p, q\} \).
In the third phase, we determine at which columns the nets in $T$ must be split; consequently runs in $L$, are split and possibly mated (as illustrated in Figs. 5-8) with runs in $S$. (Note that this heuristic mates long and short runs, thereby reducing the maximum length.)

We now show that this problem can be transformed into a min-cost max-flow problem in a suitably defined direction graph $G_F = (V', E')$. The vertex set $V'$ of $G_F$ is partitioned into five sets: $\{s\}, \{t\}, J, A,$ and $B$. Node $s$ is the source and node $t$ is the sink. Node set $J$ is a collection of jogging requests, that is, the cardinality of $J$ equals the number of target nets obtained in the previous phase of the techniques. Node set $A$ is the set of nodes chosen as possible candidates for jogging; more precisely, there is a one-to-one correspondence between nets in $T$ and nodes in $A$. Each node of $B$ is a set of columns, which represent resources to be utilized to satisfy jogging requests: specifically each node in $B$ is associated either to an empty column or to the set of terminals of a run in $S$, (short runs). The nodes in $B$ are weighted: empty-column nodes have zero weight, while a node corresponding to a run in $S$, has a weight equal to the run length. All arcs in $E'$ have capa-
In the first phase, the listing of the nets runs in time $O(c)$ and the computation of $f$ runs in time $O(cf)$ [see 16]. In the second phase, selection of the target nets is accomplished in time $O(n)$; similarly, the minimization of the horizontal overlap between target nets is done in time $O(n)$, because this task can be accomplished in time quadratic in the size of $T$ (which is $O(\sqrt{n})$). In the third phase, finding the minimum cost max-flow is accomplished in time $O(n\sqrt{n})$ because the selected algorithm [17, 18] has running time which is cubic in the size of $T$. Finally, rearrangement of runs is completed in time $O(n)$.

More complex is the analysis of the running time of the fourth phase, because in [10] the performance of the algorithm is not evaluated. To remove this obstacle, we can imagine a simpler ver-
city 1 and a cost; for \( v_1, v_2 \in V, E' \) consists of the following arcs \((v_1, v_2)\):

\[=
\begin{align*}
a) & \; v_1 = B, v_2 \in J, \text{ of cost 0.} \\
b) & \; v_2 = B, v_2 = t, \text{ of cost 0.} \\
c) & \; v_1 \in N, v_2 \in A, \text{ where } v_1 \text{ identifies a jogging request whose target net is } N, \text{ and } v_2 \text{ is either } N \text{ or a net } N' \text{ in } \text{NEIGHBOR}(N); \text{ the cost of } (v_1, v_2) \text{ is } d(N, N'). \\
d) & \; v_j \in A, v_2 \in B, \text{ where the net associated with } v_j \text{ intersects at least one of the columns associated with } v_2; \text{ the cost of } (v_1, v_2) \text{ equals the weight of } v_2.
\end{align*}
\]

It is clear that a flow of value \( j \) identifies \( j \) joggings, since it corresponds to \( j \) paths from \( s \) to \( t \) which are vertex-disjoint in \( V, ps 10(s, t) \). The min-cost criterion, and the selection of arc costs is the heuristic that matches joggings and columns.

After finding the minimum cost max-flow, runs in \( L_1 \) and in \( S_1 \) are properly rearranged as outlined in the previous section. Fig. 10(a) gives \( G' \) for the example of Fig. 1. Bold arcs indicate the solution found by the algorithm. Fig. 10(b) shows the resulting \( G' \) where the initial part of \( r_1 \) is mated with \( r_3 \) at column 6, the central part of \( r_1 \) is mated with \( r_4 \) at column 17, the final part of \( r_1 \) is mated with \( r_5 \) at column 22, and \( r_2 \) is split at the empty column 13. The choice of the jogging column for the first jogging of \( r_1 \) is made on the basis of the density.

In the fourth phase horizontal tracks are assigned to individual nets and the proper routing is determined for all the nets. The channel routing can be found by using the algorithm proposed in [10] where run nets are merged so as to minimize the length of the longest path length in the vertical constraint graph.

4. Analysis of performance

To analyze the performance of the algorithm, we shall evaluate the running time of each phase separately. Suppose that a channel has \( c \) columns, \( n \) nets and flux \( f \). Furthermore, note that the size of \( T' \) (the set of target nets) is \( O(\sqrt{n}) \).
Summing up, we conclude that the running time of the algorithm is dominated by the fourth phase and does not exceed $O(n^2)$.

We must point out, however, the following shortcomings of the current implementation of the algorithm:

- runs in the form of cycles are not treated;
- only two-terminal-net problems are considered;
- the only kind of jogging allowed is the one shown in Fig. 6 (a);
- set $B$ consists only of columns inside the channel.

It is clear that the quality of the solution given by the algorithm depends on the choice of the parameter $m$ and on how well resources in $B$ can satisfy the jogging requests of target nets. Experimental results show routings with width $t \cdot d \leq t \leq d + m$, when $d \geq (3/2)m$ and the longest run obtained at the end of Phase 3 has length very close to $m$. Fig. 11 shows the channel routing obtained for the problem of Fig. 1: it has optimal width and it makes four joggings.

Fig. 11. An optimal routing obtained by the proposed algorithm for the channel problem of Fig. 1.
References


