2.25 Manhattan routing

[CLRS]

Problem 4-2. Manhattan Channel Routing

A problem that arises during the design of integrated-circuit chips is to hook components together with wires. In this problem, we'll investigate a simple such problem.

In ***Manhattan routing***, wires run on one of two layers of an integrated circuit: vertical wires run on layer 1, and horizontal wires run on layer 2. The height h is the number of horizontal tracks used. Wherever a horizontal wire needs to be connected to a vertical wire, a ***via*** connects them. Figure 1 illustrates several ***pins*** (electrical terminals) that are connected in this fashion. As can be seen in the figure, all wires run on an underlying grid, and all the pins are collinear.

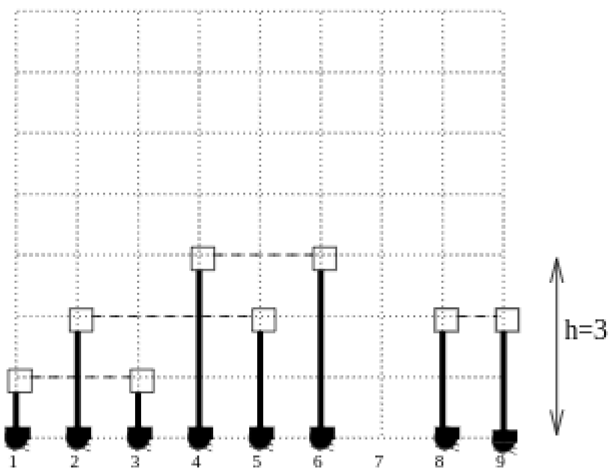

Figure 1: Pins are shown as circles. Vertical wires are shown as solid. Horizontal wires are dashed. Vias are shown as squares.

In our problem, the goal is to connect up a given set of pairs of pins using the minimum number of horizontal tracks. For example, the number of horizontal tracks used in the routing channel of Figure 1 is 3 but fewer might be sufficient.

Let $L = \{(p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)\}$ be a list of pairs of pins, where no pin appears more than once. The problem is to find the fewest number of horizontal tracks to connect each pair. For example, the routing problem corresponding to Figure 1 can be specified as the set $\{(1, 3), (2, 5), (4, 6), (8, 9)\}$.

(a) What is the minimum number of horizontal tracks needed to solve the routing problem in Figure 1?

Solution: You can verify that the wire connecting pins 4 and 6 could be at the same height as the wire connecting pins 1 and 3, making the number of horizontal track needed 2. Note that this is the minimum possible. Otherwise the wire connecting pins 2 and 3 and the wire connecting 1 and 3 would be on the same track, violating the problem specifications.

(b) Give an efficient algorithm to solve a given routing problem having n pairs of pins using the minimum possible number of horizontal tracks. As always, argue correctness (your algorithm indeed minimizes the number of horizontal tracks), and analyze the running time.

Algorithm description

The following algorithm routes pin pairs greedily into available horizontal tracks in order of the smaller pin of a pair.

1. Go through L and if $q_i > p_i$ swap them. For each pair, we call the smaller pin, start, and the larger, end. Sort the start and end pins of the pairs in an increasing order. The resulting list contains $2n$ values.
2. Place all available horizontal tracks in a stack S.
3. Go through the list in sorted order.
   - If the current pin is a start pin, pop the first available horizontal track from S and route it in that horizontal track. If S is empty, then it is not possible to route all of the pin pairs using the given number of horizontal tracks. Report an error in this case.
   - If the current pin is an end pin, look up in which horizontal track it has been routed and push that horizontal track back onto S.

Correctness

Suppose that the algorithm terminates with a routing requiring m horizontal tracks. Let k denote the first pair of pins routed in the mth horizontal track. Let s_k denote the start pin and f_k denote the end pin of this pair. Let f_l be the earliest finish pin appearing in the sorted list after s_k. Necessarily, $f_l > s_k$. The closest routing in each of the $m - 1$ horizontal tracks already in use starts before s_k. Each routing terminates after f_l. Thus there are m routings in between $[s_k, f_l]$, i.e., any routing must use at least m vertical tracks. Thus the routing returned by the algorithm is optimal.

The above argument shows that given an infinite supply of horizontal tracks, our algorithm will always produce a routing that uses the fewest number of horizontal tracks. Thus, if the algorithm terminates with an error, it means that a given number of horizontal tracks is less than the number of horizontal tracks in an optimal routing, and hence it is impossible to route all the pin pairs.

Analysis

This algorithm runs in $O(n \lg n)$ because it is necessary to sort $2n$ items (which can be accomplished using heapsort or mergesort). Notice that scanning through the list and assigning horizontal tracks takes $O(1)$ time per connection, for a total of $O(2n) = O(n)$ time.

This is also known as the ***interval-graph coloring problem***. We can create an interval graph whose vertices are the given pairs of pins and whose edges connect incompatible pairs of pins. The smallest number of colors required to color every vertex so that no two adjacent vertices are given the same color corresponds to finding the fewest number of horizontal tracks needed to connect all of the pairs of pins.)