Fast Algorithms for One-Dimensionsal Compaction with Jog Insertion

Matthias F.M. Stallmann[1] and Thomas A. Hughes[2]

[1] Dept. of Computer Science, North Carolina State University, Raleigh, NC 27695
[2] IBM Corp., Dept. D63/Bldg. 061, Box 12195, Research Triangle Park, NC 27709

Abstract. Subquadratic algorithms are given for each of two phases of channel compaction: determining minimum channel height and producing a minimum height routing that minimizes length and number of jogs for each wire. The algorithms use balanced trees with dynamic finger searching.

1 Introduction

Input to the one-dimensional channel compaction problem with automatic jog insertion consists of n horizontal wire segments organized into t tracks. Segments on the same track have the same y-coordinate and the ranges of their x-coordinates do not overlap. Each segment has a via at each end, connecting it to vertical wires on another layer. The goal is to minimize channel height subject to design rule constraints on the distances between features (wires and vias). The relative vertical position of wires is not allowed to change during compaction.

Figure 1(a) shows an initial layout with 7 tracks. The horizontal wires to be compacted are shown as thin solid lines while vertical wires on another layer are thick dashed lines. Dots represent vias. Relative vertical position must be maintained due to *vertical constraints* (see e.g. [12]). The left part of net A, for example, must be above net B because the vertical wires of these nets share the same x-coordinate. The right part of net A must be below net B for the same reason. It is these vertical constraints that prevent the problem from being solved by interval graph coloring.

Figure 1(b) shows the result of a greedy compaction — each via and wire segment is pushed as low as possible, subject to the constraint that it is at least unit distance away from any other via or wire segment (in the Manhattan metric). To keep our presentation simple, we assume that vias are the same width as wires. The algorithms can be modified to handle the more typical case where vias are wider than wires (see Section 5).

Greedy compaction often introduces unnecessary *jogs* or wire bends. Figure 1(c) shows the result of a wire-straightening heuristic that minimizes the number of jogs for each wire in turn, beginning with the uppermost track. While this approach does not necessarily minimize the total number of jogs, the number of jogs is reduced significantly from that of greedy compaction.

We give the fastest known algorithms for each phase of a two-phase compaction process. Phase 1 determines minimum channel height by producing (an implicit representation of) a greedy compaction. Phase 2 produces a minimum-height layout in which the number of jogs for each wire is minimized. Let n be the number of horizontal wire segments in the input routing (which has no jogs).

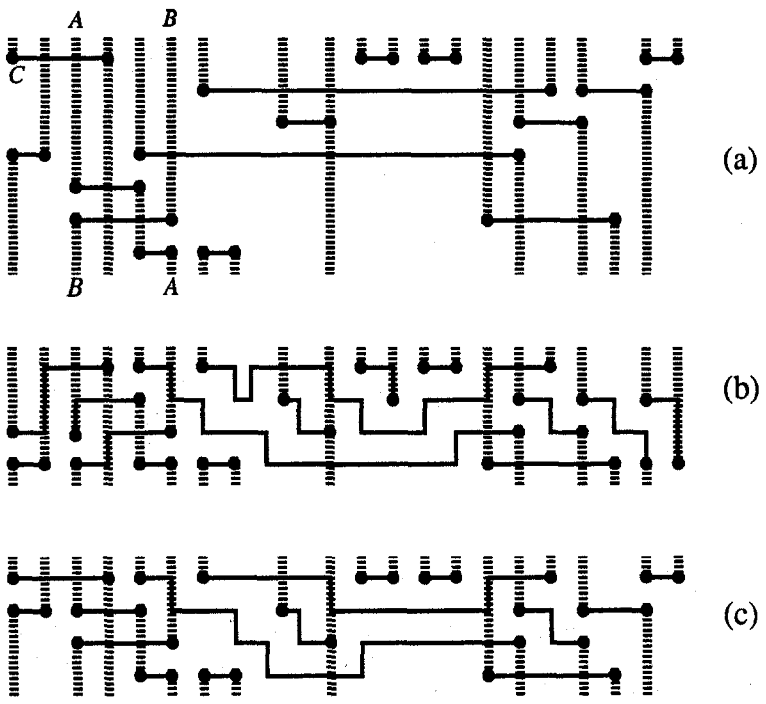

Fig. 1. An example showing two phases of compaction.

Let t be the number of tracks in the input, and let k_G and k_O be the number of wire segments in the greedy compaction of the input and the final output, respectively. Our algorithm for phase 1 runs in time $O(n \log t)$, while phase 2 requires $O(n \log t + k_O \log(k_G/k_O))$. This can be improved to $O(n \log(k_G/n))$ for phase 1 and $O(k_O \log(k_G/k_O))$ for phase 2 if the input is assumed to be *precompacted*, that is, no wire can be moved into a lower track without conflicting with other wires. In Figure 1(a), segment C is not precompacted because it can be moved to the next lower track. The latter bound is always at least as good since nt is a trivial upper bound on the number of greedy jogs. If the input is not precompacted, $\Omega(n \log t)$ is a lower bound for any algorithm that computes a routing — a special case is merging t sorted lists containing a total of n nonoverlapping wires.

The best previous bounds were $O(k_G)$ for phase 1 [17] and $O(k_G)$ for phase 2 [16] (add $n \log t$ if the input is neither sorted by x-coordinate nor precompacted). Simple examples show that k_G is in $\Omega(nt)$. Our bounds are achieved using a balanced tree with finger searching to maintain the minimum achievable height at each x-coordinate.

It appears that our phase 1 algorithms will work for distance norms other that the standard rectilinear (Manhattan) norm, allowing wire segments that are not necessarily horizontal or vertical. For example, *curvilinear* wiring allows wire segments to be straight lines or circular arcs and distances can be measured