A Faster One-Dimensional Topological Compaction Algorithm with Jog Insertion

Hsiao-Feng Steven Chen*and D. T. Lee†

Department of Electrical and Computer Engineering
Northwestern University
Evanston, Illinois 60208.
Tel: (847)491-5007
E-mail: dtlee@ece.nwu.edu

Abstract

We consider the problem of one-dimensional topological compaction with jog insertions. By combining both geometric and graph theoretic approaches we present a faster and simpler algorithm to improve over previous results. The compaction algorithm takes as input a sketch consisting of a set F of features and a set W of wires, and minimizes the horizontal width of the sketch while maintaining its routability. The algorithm consists of the following steps: constructing a horizontal constraint graph, computing all possible jog positions, computing the critical path, relocating the features and reconstructing a new sketch homotopic to the input sketch, which is suitable for detailed routing. The algorithm runs in $O(|F| \cdot |W|)$ worst-case time and space, which is asymptotically optimal in the worst case. Experimental results are also presented.

*Current address: Avant! Corporation, 46871 Bayside Parkway, Fremont, CA94538, e-mail: schen@avanticorp.com.
†Supported in part by the National Science Foundation under the Grants CDA-9703228 and CCR-9731638, and by the Office of Naval Research under the Grants No. N00014-95-1-1007 and No. N00014-97-1-0514.

1 Introduction

Layout compaction is the last stage of VLSI design whose main objective is to minimize the chip area. The algorithm that realizes this task is called a *compactor*[Ohts 86, Leng 90]. Since it is usually performed after the stages of placement and routing, the global route topology and the relative positions of the modules of the input layout provided for the compactor is assumed to be fixed. That is, during compaction the topology of wires should stay the same and there should be no drastic changes such as swapping of modules or features. Keeping this rule in mind, the compactor moves the modules or wires according to the spacing constraints dictated by technology-dependent design rules and generates a layout with a minimum area. The general 2-D compaction, which is to minimize the entire chip area, is known to be NP-hard[SaPa 82], so a number of heuristics were proposed[WoMa 88] [ShLo 89] [SLW 83]. In this paper we consider only 1-D compaction, which is aimed to minimize either the horizontal or vertical *width* of the chip[LoVa 90] [Dun 78].

In general, the 1-D compaction problem is formulated as a graph-theoretic problem by transforming the layout into a horizontal constraint graph[1] and computing its critical path[WoMa 88, KeWa 84]. The non-horizontal wire segments associated with a net can be treated as rigid objects(Fig. 1(a)) so compaction of objects is constrained by these wire segments, or they are considered flexible and are allowed to bend when the modules are being compacted. In the latter case the compaction is called the *topological compaction*[Leng 90] or *homotopic compaction*[Male 91, MeNa 90], and the bends are called *jogs*. (See Figure 1.) The best known algorithm for graph-based 1-D compaction (without jog insertions) runs in

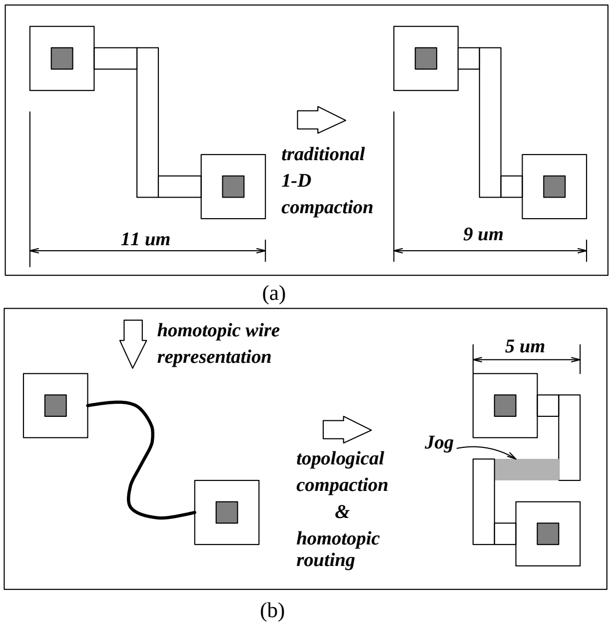


Figure 1: Traditional 1-D compaction versus topological compaction

$O(N^{1.5} \log N)$ time[LoVa 90], and those for the geometry-based 1-D compaction problem (with jog insertions) run in $O(N^4)$[Male 91] and $O((N^2+k) \log N)$ time[MeNa 90] respectively, where N is the input size, and k is some measure of the differences between the input and output topology which may be $O(N^3)$. Gao et al. also proposed an $O(N^2 \log N)$ time algorithm for the topological compaction with a non-trivial implementation for both planar and knock-knee routing model[Gao 89]. Although the topological com-

[1] From now on we assume that the horizontal width of the chip is to be minimized.

paction algorithms are proposed for only one-layer routing model, extensions to multi-layer layout model can be done by applying these algorithms to every layer at the same time[MeNa 90]. Moreover, it could also be applied to general-architecture layout style used in sea-of-gate chips, where global routing breaks the chip area into a uniform two dimensional array of global routing cells (GRC) and then detailed routing is performed in the GRCs[LiMa 84, ChSa 90, Card 91, Kao 95]. Unlike the channel-based layout design, after the modules or the standard cells are placed in this kind of general-architecture layout style, there is only little flexibility controlling the congestions of the bottleneck routed area. Sometimes, slightly pushing a module can improve the utilization of some routing area with insufficient capacity to accommodate more nets. The idea of pushing modules was first introduced by the *plowing* concept in Magic Layout Editor used for interactively changing the geometry of the layout[Oust 84, Sott 84]. A plowing with jog insertion can make use of the area more efficiently than the conventional plowing. The reader may find more topological compaction applications in [MeNa 90, Leng 90].

Generally speaking, the basic steps of a topological compactor are as follows: Firstly, the layout topology is described by a *sketch* model represented by the *features* or *modules* and flexible *wires*. The wires are usually transformed into a normal form, called *rubber-band equivalent(RBE)*[LeMa 85, CoSi 84, DaDa 91, DaKo 90, ChLe96]. The second step is to determine the moving ordering and the displacement of the modules or features, while maintaining the relative positions of these modules, and the so-called *safe* routing condition for wires[CoSi 84, LeMa 85]. If the routability is assured by the compactor, the final sketch is considered legal and the precise geometry of the detailed routing can be obtained without violating design rules. Finally, a *homotopic router* is employed to reconstruct the detailed routing for the nets according to the topology of the wires, called *homotopy*[CoSi 84, LeMa 85] and the required routing model, i.e., single-layer[LeMa 85] in grids or gridless, knock-knee model[KaMe 94], etc. The rest of the paper is organized as follows. We first provide some definitions and common terminologies in Section 2. We then explain how the routability condition is used as a gauge to form the spacing constraints in Section 3. We describe in Section 4 our topological compaction algorithm. and analyze the overall time complexity in Section 5. In Section 6 we discuss extensions to multi-layer and multi-terminal nets layout. We conclude with some experimental results.

2 Preliminaries and Previous Work

2.1 Components of a Sketch

The input to a homotopic router or a topological compactor is usually represented by a *sketch*[LeMa 85, MeNa 90]. As is shown in Fig. 2(a), a sketch $S = (F, M, W)$ is a planar drawing of a physical layout in the grid, where F is a finite set of *features*, which are points or open line segments on the boundaries of the modules m_i's in M, and W is a finite set of *wires*, which are vertex-disjoint simple paths connecting the point features by a sequence of straight line segments (Fig. 2). To simplify the discussion, wires are assumed to have exactly two endpoints (point features) which are called *terminals*, and all the features lie on a grid.

2.2 Homotopy of Wires, and Detailed Routing in Topological Compaction

A detailed routing of a sketch $S(F, M, W)$, $W = \{\mu^1, ..., \mu^n\}$, is a sketch $S^*(F, M, W^*)$, $W^* = \{\nu^1, ..., \nu^n\}$, such that ν^i is *homotopic* to μ^i, i.e., net μ^i and net ν^i have the same pair of terminals, and μ^i can be

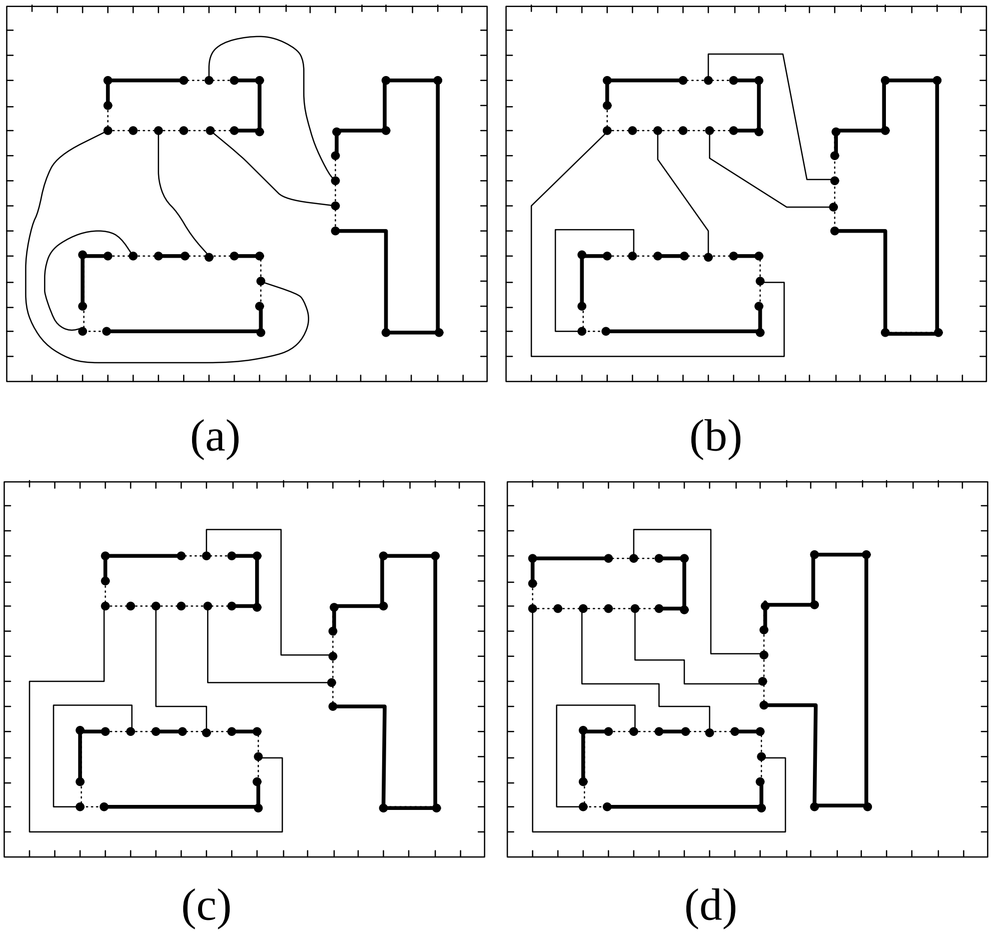


Figure 2: Sketches (a) S (b) $\overline{S}$ (c) S^* (d) S^c

transformed continuously into ν^i without intersecting any feature in F, W^* optimizes both individual and total wire length, and the ν^i's in W^* are vertex-disjoint paths in the rectangular grid of unit-spacing, where n represents the total number of nets. The detailed routing S^* of the sketch S in Figure 2(a) is shown in Figure 2(c). Figure 2(b) shows the associated VRBE $\overline{S}$ and Figure 2(d) shows the sketch S^c after topological compaction. Note that with conventional 1-D compaction, the sketch S can only be compacted by one unit (Fig. 2(c)), while with topological compaction it can be compacted by three units (Fig. 2(d)).

The homotopy of a net μ^j in W is specified by a sequence of wire segments, denoted as $\mu^j =< s_0^j, s_1^j, ..., s_l^j >$ for some l, where (s_k^j, s_{k+1}^j) denotes wire segment w_k^j. The size of the wire homotopy is the total number of wire segments, denoted as $|W| = \sum_j \sum_k w_k^j$. These wire segments are the basic units adopted in [ChLe96] for obtaining a vertex-disjoint RBE and detailed routing. The following result is obtained in [ChLe96].

Lemma 1 *Given a sketch $S(F, M, W)$, its vertex-disjoint rubber-band equivalent sketch can be computed in time $O(|F| \cdot |W|)$ in the worst case.*

2.3 Routability Condition and Topological Compaction

Before the final detailed routing is invoked, it is necessary to check if the wires going through the channels between the modules are properly assigned by the topology, especially after changing the position of some feature. That is, we need to check if there is any channel with insufficient space for the wires to pass through. Such legality test is also crucial for the compaction problem. We shall follow the terminology given in [LeMa 85]. A sketch is *legal* if (1) every feature lies on the grid, and (2) the wires in a sketch are vertex-disjoint paths on the grid. Let $c = (x(c), y(c))$ and $d = (x(d), y(d))$ be two points in F, and let $\overline{cd}$ denote the open line segment between c and d. A *cut* is an open line segment, which intersects no features in F. In other words, points c and d are *visible* to each other. A cut $\overline{cd}$ is *critical* if either (1) both c and d are point features, or (2) c is a point feature, and d is the point on a line feature closest to c. The *capacity* of a cut is defined as the *maximum* number of legal wires that can cross the cut. More precisely, the capacity of a cut $\overline{cd}$, is given as $cap(\overline{cd}) = \max\{|x(c) - x(d)|, |y(c) - y(d)|\} - 1$, or $\max\{\Delta x, \Delta y\} - 1$. The actual number of wires of the sketch crossing a cut is called the *density* of the cut. A cut is said to be *safe*, if its density is not greater than its capacity, and *saturated* if they are equal. The following lemma is obtained in [CoSi 84] and [LeMa 85].

Lemma 2 *A sketch $S = (F, M, W)$ is routable if and only if $(F, M, \emptyset)$ is legal and every critical cut in (F, M, W) is safe.*

A *diagonal cut* $\overline{cd}$ is a cut emitted from point or corner feature c to d with slope $+1$ or -1, Associated with each point feature f_i there are at most four diagonal cuts. In the following, we use $f_i(\theta)$, $\theta = k\pi/4, k = 1, 3, 5, 7$, to denote the four diagonal cuts with respect to feature f_i in θ direction.

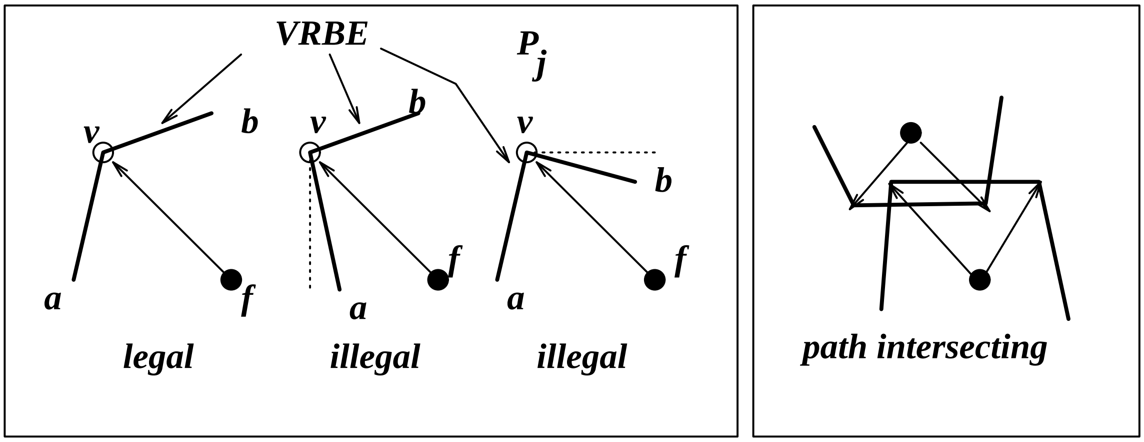
In the final routing stage, in order to maintain the legality of the layout (safeness), the precise distance of wires from the corner features, against which a number of paths are tightened resulting from the VRBE transformation, must be computed. More precisely, let f denote a corner feature and w the kth wire touching against f. Then w must cross a diagonal cut $f(\theta)$ at a point z' whose distance $|\overline{fz'}|$ from f is at least $|\overline{fz}| = \sqrt{2}k$. The segment $\overline{fz}$ with respect to wire segment w is called an *obstacle spoke* for w, and z is a *supporting vertex* of w. A spoke is positive if its slope is $+1$, and negative if -1.

Using Lemma 2, the authors of [Male 91] and [MeNa 90] proposed algorithms for the topological compaction problem. More specifically it is shown in [LeMa 85] that transformation of the sketch into its RBE form takes $O(|F| \cdot |W| \log |F|)$ time and $O(|F| \cdot |W|)$ space. Compaction can then be performed on the RBE representation in $O(N^4)$ [Male 91] and $O(N \cdot |W| + (N^2 + O(|F|^3)) \log N)$ [MeNa 90] time, respectively, where $N = |F| + |W|$.

Based on Lemma 2, Chen and Lee[ChLe96] recently proposed an algorithm to transform a sketch into a vertex-disjoint RBE and provided a new routability condition as follows.

Lemma 3 *Given a sketch $S(F, M, W)$, an unsafe critical cut exists if and only if there exists an illegal vertex in some net $\mathcal{N}^j \in \overline{W}$ of its VRBE $\overline{S}(F, M, \overline{W})$ or there exist intersecting paths (Fig. 3).*

An internal vertex of $\mathcal{N}^j$ in $\overline{W}$ is *illegal* if the angle formed by the net and the spoke supporting the RBE is less than $\pi/4$. These conditions are shown in Fig. 3(a), where v is a supporting vertex with respect to spoke $f(3\pi/4)$. v is illegal if either angle $\angle fva$ or $\angle fvb$ is less than $\pi/4$, resulting in an unsafe sketch. Furthermore when there is some pair of nets intersecting each other as shown in Fig. 3(b), the corresponding cut is not routable also. The above Lemma not only provides a good test for routability but also a good way to compute the exact spacing constraints between the vertices corresponding to the features in the constraint graph associated with the layout sketch. Intuitively the wires in each net shouldn't be bent too much at any vertex to make it illegal or moved too close to make the nets intersecting.



(a) (b)

Figure 3: Two unsafe conditions in vertex-disjoint RBE.

Denote the displacement of a feature $f \in F$ by $\vec{d}(f)$ and call the vector $\mathbf{d} \in \mathbf{R}^{|F|}$ of displacements a *configuration*. To maintain the same relative topology as in the given sketch S, the following two conditions must hold: (1) $x(p) + \vec{d}(f) < x(q) + \vec{d}(g)$ for any $p = (x(p), y(p))$ and $q = (x(q), y(q))$ where $x(p) < x(q)$ and $y(p) = y(q)$ and p lies on a feature f and q lies on a feature g, and (2) $\vec{d}(f) = \vec{d}(g)$ for any two features in the same module. The compacted sketch is denoted by $S(\mathbf{d})$ and $S(\mathbf{d}) = (F(\mathbf{d}), M, W^c)$ such that a point p on feature f with coordinates $(x(p), y(p))$ in S has coordinates $(x(p) + \vec{d}(f), y(p))$ in $S(\mathbf{d})$ and the wires W^c have the same global routes in S. Let $\mathcal{C}(S) \subseteq \mathbf{R}^{|F|}$ be the *configuration space* that satisfies the above topology conditions. The topological compaction problem can be formalized as the following:

> Given a routable sketch $S = (F, M, W)$, find a configuration $\mathbf{d} \in \mathcal{C}(S)$ such that $S(\mathbf{d})$ has minimal x-width.

The following theorem is the main result of this paper.

Theorem 1 *The one-dimensional topological compaction problem can be solved in time $O(|F| \cdot |W|)$, which is asympototically optimal in the worst case.*

2.4 Horizontal Constraint Graph and Jog Insertion

Traditionally in a graph-based 1-D compaction algorithm, a horizontal constraint graph $G(V, E)$ is used to describe the minimal horizontal spacing requirements between the left and right vertical edges of any two horizontally visible rigid objects. We assume for simplicity the minimal wire-to-feature, wire-to-wire and feature-to-feature spacing is one unit and the width of a wire is zero.

Lemma 4 *The horizontal constraint graph $G(V, E)$ associated with a sketch S of N objects ($N = |F| + |W|$) can be computed in time $O(N \log N)$. Moreover, $|V(G)| = N$ and $|E(G)| = O(N)$, and the longest path can be computed by topological sorting in time $O(|F| + |W|)$.*

Fig. 4(a)(b) shows a sketch S and its horizontal constraint graph G, (c) shows the result after conventional 1-D compaction in L_1 metric. Since the length of the longest path (i.e., $\mathbf{L} \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow \mathbf{R}$) is five in (b), the width of the final layout is five unit.

We now consider topological compaction. Recall that $\overline{S}(F, M, \overline{W})$ denotes the VRBE sketch, and a wire segment w_k^j in net $\mathcal{N}^j = < s_0^j, s_1^j, \ldots, s_l^j >$ for some l is defined by the x and y coordinates of the two end points, i.e., $w_k^j = (s_k^j, s_{k+1}^j) = ((x_k^j, y_k^j), (x_{k+1}^j, y_{k+1}^j))$. Note that $|\overline{W}|$ could be $O(|F| \cdot |W|)$[ChLe96]. The wire segments in each net $\mathcal{N}^j$ will be partitioned into two subsets, H^j and V^j, consisting of horizontal segments and non-horizontal segments respectively. The non-horizontal wire segments of $\mathcal{N}^j$ are further decomposed into $\overline{C^j}$, a collection of *y-monotone chains* $\mathcal{P}_t^j$ for some t, i.e.,

$$\overline{C^j} = \bigcup_t \mathcal{P}_t^j$$

Definition 1 (Fig. 5(a)) *A y-monotone chain $\mathcal{P}_t^j$ is a collection of consecutive wires which is a subset of the VRBE net $\mathcal{N}^j$ for some t such that no two internal points p and q in $\mathcal{P}_t^j$ are of the same y-coordinate.*

Let $\overline{C}$ denote the collection of $\overline{C^j}$ and $\overline{H}$ denote the collection of $\overline{H^j}$ for all net $\mathcal{N}^j, 1 \leq j \leq n$. That is, $\overline{W} = \overline{H} \cup \overline{C}$:

1. Horizontal wires:
$$\overline{H} = \{w_k^j \ : \ |y_k^j - y_{k+1}^j| = 0, \ 0 \leq k < l\}$$

2. Nonhorizontal wires:
$$\overline{C} = \{\mathcal{P}_t^j : |y_k^j - y_{k+1}^j| \neq 0, \ 0 \leq \alpha \leq k < \beta \leq l\}$$
   where
$$\mathcal{P}_t^j = (w_\alpha^j, \ldots, w_\beta^j)$$
   and y_α^j and y_β^j are the minimal and maximal y-coordinates, respectively over y_k^j.

Fig. 5(a) shows an $\overline{S}$ and the y-monotone chains $\overline{C}$ are depicted in bold lines while the horizontal wire portions $\overline{H}$ are shaded. The number of y-monotone chains with respect to a net $\mathcal{N}^j$ is determined by the VRBE homotopy. For example, the number of y-monotone chains in net $\mathcal{N}^j$ in Fig. 5(a) is three, i.e., $t = |\overline{C^j}| = 3$. The following Lemma is obvious:

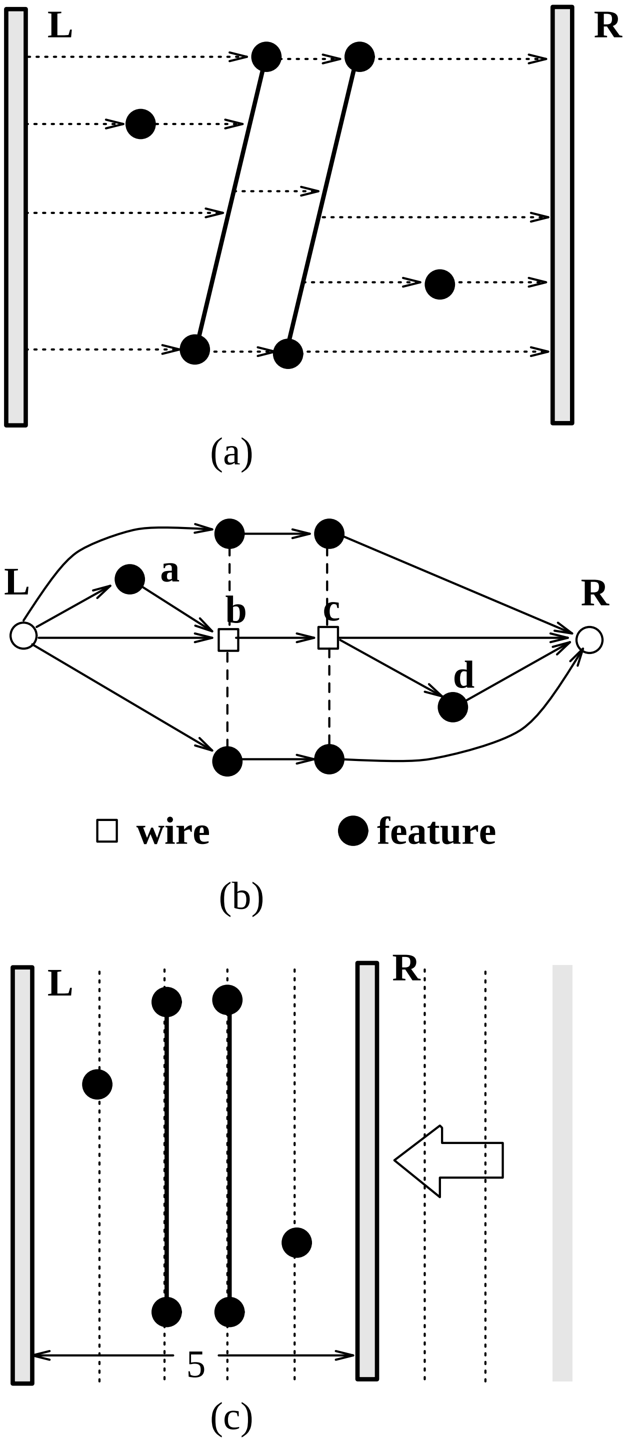
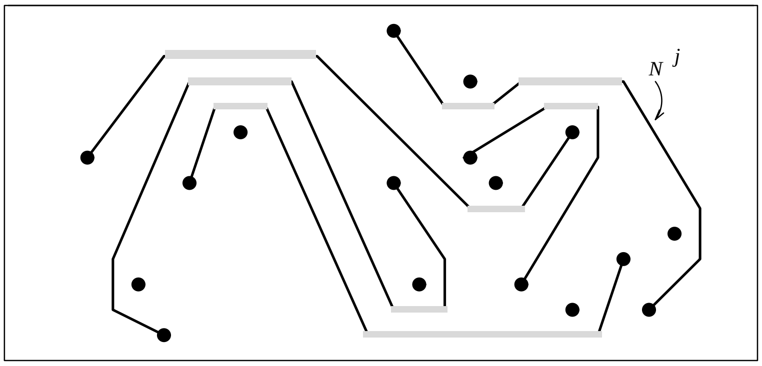
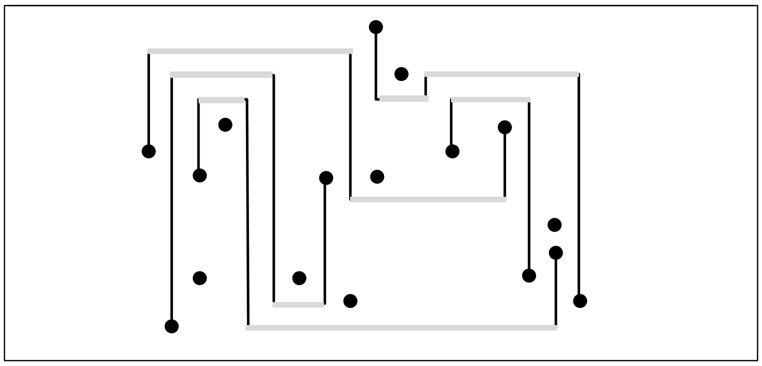


Figure 4: (a) a sketch (b) horizontal constraint graph (c) after compaction.

Lemma 5 *Given a sketch $S(F, M, W)$, the total number of y-monotone chains in VRBE $\overline{S}$, denoted $|\overline{C}|$, satisfies $|\overline{C}| \leq |W|$.*



(a)



(b)

Figure 5: (a) y-monotone chains in a VRBE sketch (bold lines) (b) the chain sketch $\overline{\overline{S}}$

Obviously, a jog can only be introduced in the y-monotone chains, and these chains will affect horizontal visibility among features. To highlight this visibility relation between the features and chains, and to ease the description of our algorithm, we introduce a *chain sketch* derived from VRBE $\overline{S}$ as follows: A chain sketch $\overline{\overline{S}}(\mathbf{d}) = (\overline{F}(\mathbf{d}), M, \overline{\overline{C}})$ is a routable sketch transformed from VRBE sketch $\overline{S}(F, M, \overline{C})$ by some $\mathbf{d} \in \mathcal{C}(S)$ such that $\overline{\overline{S}}$ has the same (horizontal) visibility as in $\overline{S}$ but every y-monotone chain $\mathcal{P}_t^j \in \overline{C}$ is replaced with a vertical line segment $I_t^j(x) = (x, y_\alpha^j, y_\beta^j) \in \overline{\overline{C}}$ for some x and the horizontal distances between every pair of objects are legal. In the following, we call $I_t^j(x)$ the *chain interval* corresponding to y-monotone chain $\mathcal{P}_t^j$ with y_α^j and y_β^j as extreme values. Fig. 5(b) shows the associated chain sketch $\overline{\overline{S}}$ for the VRBE sketch in (a). Note that in $\overline{\overline{S}}$, the y-coordinates y_α^j and y_β^j for each $I_t^j(x)$ are equal to those of chain $\mathcal{P}_t^j$ but the x-coordinates are displaced so that $\overline{\overline{S}}$ is legal and the relative visibility information from VRBE $\overline{S}$ is maintained. Since $\overline{C}$ to $\overline{\overline{C}}$ is a one-to-one mapping, $|\overline{C}| = |\overline{\overline{C}}|$. We will show how the constraint graph G associated with chain sketch $\overline{\overline{S}}$ can be computed from a given VRBE sketch $\overline{S}$ in a later section.

With the y-monotone chain concepts, let us consider Fig. 6 and its constraint graph. Recall that Fig. 4(b) is generated by a conventional 1-D compaction and the critical path determines the compacted width: in this case, as described, the longest path between $\mathbf{L}$ and $\mathbf{R}$ is $\mathbf{L} \to a \to b \to c \to d \to \mathbf{R}$, so the length is five, which means the minimal width is five. Note that all the y-monotone chains are treated as rigid objects and cannot be bent during compaction. On the other hand, in topological compaction, the non-horizontal y-monotone chains are treated flexible and could be deformed into many pieces (i.e., broken vertical lines form jogs) to shorten the width. Fig. 6(a) shows that if we can somehow identify

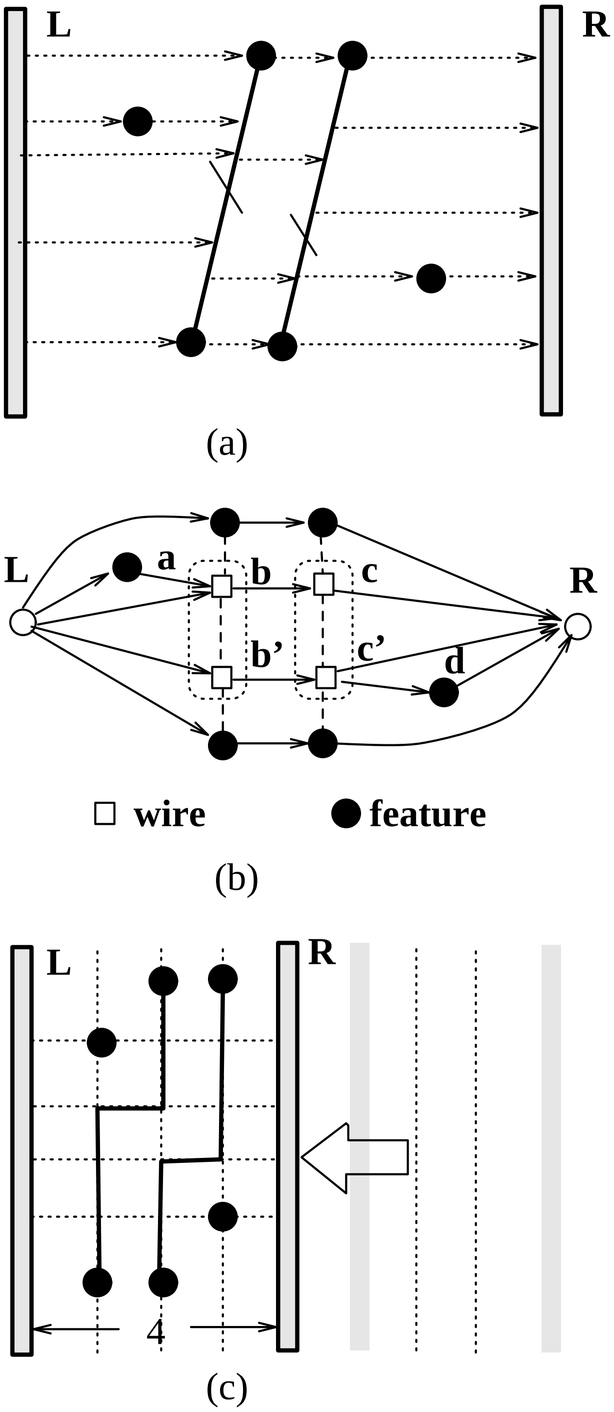


Figure 6: (a) A sketch (b) constraint graph (c) after compaction.

the position(s) to split the vertices associated with the y-monotone chains, further compaction is possible by breaking one longer critical path into two shorter ones (Fig. 6(b)): As shown in Fig. 6(a), the original longest path is divided into two by splitting off new vertices b' and c' from the original vertices b and c; $\mathbf{L} \to a \to b \to c \to \mathbf{R}$ and $\mathbf{L} \to b' \to c' \to d \to \mathbf{R}$. Fig. 6(c) shows the final compacted result. Both paths are of length four, which is better than that in Fig. 4(c). However, if we move up or down one of the two isolated point features one unit closer (change their vertical distance from three to two), the wires will not be allowed to bend. Otherwise, the layout is illegal based on Lemma 2 or Lemma 3. In our graph-theoretic approach for topological compaction, such a vertex splitting is a crucial step. The idea of breaking the longest path in the constraint graph is not new. The vertex splitting idea has been proposed before but only used as a heuristic to iteratively break the critical path without a systematic method to manipulate the jogs[WoMa 88] and as a result, it need not necessarily produce an optimal solution. Our main contribution is to provide a systematic method to introduce jogs breaking the longest path and obtaining an optimal width after compaction.

3 Spacing Constraints in a Sketch

In this section, we explain how the spacing constraints in a VRBE sketch are formulated according to Lemma 3. Consider an example shown in Fig. 7 containing a point feature f with an obstacle spoke f^1 in the direction $\theta = 3\pi/4$ of length $\sqrt{2}$ and three nets. Imagine that we move feature f toward left. f^1 will

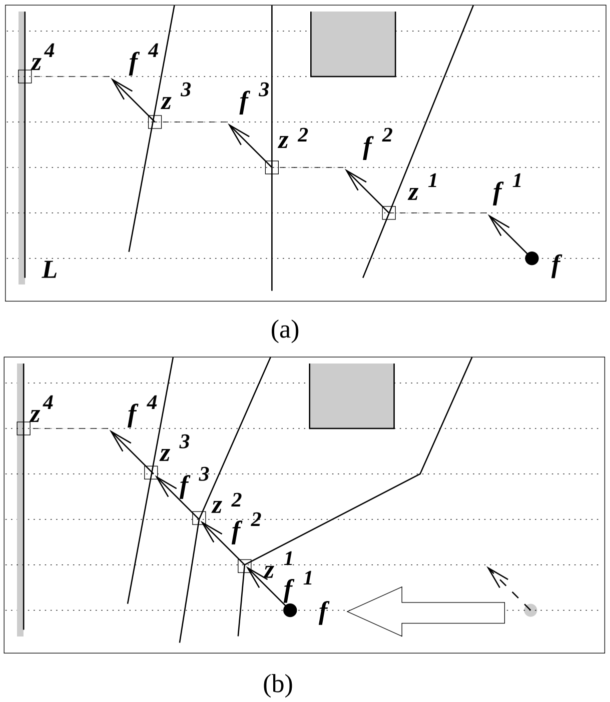


Figure 7: P-spokes in $3\pi/4$ direction

hit the first net at vertex z^1 as indicated. The spoke f^1 is now translated to become spoke f^2. Continuing the movement of feature f toward the left, the first net will be bent and f^2 will hit a vertex z^2 on the second net. In so doing we obtain spokes f^1, f^2, f^3 and f^4 and vertices z^1, z^2 and z^3. This process ends when spoke f^4 hits a line feature at vertex z^4. The vertices z^i on the nets are the *potential supporting vertices*, or *jog vertices*, at which the nets can be bent, when the features are being compacted. The spokes

f^1 through f^4 will be referred to as P-spokes. Note that the above jog vertex creation process is defined in a chain sketch where all the nets are replaced by the vertical chain intervals. We define P-spokes and jog vertices in a chain sketch formally below. Let $\theta \in \{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}\}$.

Definition 2 (P-spoke, and jog vertex) *A P-spoke of rank 1 associated with a feature f in direction θ, denoted $f^1(\theta)$, is an obstacle spoke $\overline{fz}$ emanating from f in direction θ such that $|y(z) - y(f)| = |x(z) - x(f)| = 1$. A jog vertex of rank 1 associated with a feature f in direction θ, denoted $z^1(f_\theta)$, is a vertex on a net in sketch $\overline{\overline{S}}$ such that it is horizontally visible from f^1. That is, $y(p) = y(z)$ and $|x(p) - x(z)| \geq 1$, where $p = z^1(f_\theta)$. A P-spoke of rank k associated with a feature f in direction θ is defined recursively as a spoke $\overline{pz}$ emanating from p in $f^k(\theta)$ associated with a feature f in direction θ such that $|y(z) - y(p)| = |x(z) - x(p)| = 1$, where $p = z^{k-1}$ is a jog vertex of rank $k-1$ associated with f. A jog vertex of rank k associated with a feature f in direction θ, denoted $z^k(f_\theta)$, is a vertex on a net in sketch $\overline{\overline{S}}$ such that it is horizontally visible from f^k. That is, $|y(p) - y(f)| = k$ and $|x(p) - x(f)| \geq k$ $p = z^k(f_\theta)$. The rank of a jog vertex p is denoted as $rank(p)$.*

We say that the jog vertex $z^k(f_\theta)$ is induced by P-spoke f^k. The lemmas below are immediate following Lemma 5:

Lemma 6 *Let p be a jog vertex z^k induced by a P-spoke associated with a feature f. Then $rank(p) \leq |\overline{\overline{C}}| \leq |W|$, where $|W|$ is the size of the initial sketch $S(F, M, W)$.*

For example, in Fig. 7(a) since there are at most $|W|$ y-monotone chains to the left and right of the feature f, we have jog vertices of rank at most $|W|$.

Lemma 7 *Given a sketch $S(F, M, W)$, let J denote the set of induced jog vertices and $\overline{\overline{C}}$ the set of chain intervals in the associated chain sketch $\overline{\overline{S}}$. Then we have (1) each chain interval has at most $|F|$ jog vertices and (2) $|J| \leq \sum_{i=1}^{|\overline{\overline{C}}|} |F| = |F| \cdot |\overline{\overline{C}}| \leq |F| \cdot |W|$.*

In the following we describe how Lemma 3 is related to the horizontal constraints in the constraint graph G. First of all, according to Lemma 3, if a sketch after compaction is unroutable, it must be caused by the existence of illegal vertices which results from some overpushed features (see the definition of illegal vertices in Fig. 3). Moreover, the illegal vertices must occur in pairs on the same net by two P-spokes in opposite horizontal directions (one toward left and one toward right) (see Fig. 8: if the two features move any closer, the two jog vertices become illegal). We shall therefore consider the relation among the induced jog vertices on the same net. In particular, we are interested in the jog vertex pairs on the same net induced by two P-spokes in opposite horizontal directions. As those jog vertices are induced along the y-monotone chains, they also provide the horizontal spacing constraints between the features. Assume B and A, $y(A) < y(B)$, are two jog vertices along some net $\mathcal{N}^j$ next to each other and are induced by spokes $f_2^{k'}(\theta)$ and $f_1^k(\theta + \pi)$ with $\theta = \frac{3\pi}{4}$ or $\frac{\pi}{4}$, respectively, as shown in Fig. 8. Accordingly, there are two possible constraint types between f_1 and f_2:

- $\lambda(\frac{3\pi}{4}, \frac{7\pi}{4})$-constraint or λ_--constraint for brevity: Formed by the spokes of negative slope, $f_2^{k'}(\frac{3\pi}{4})$ and $f_1^k(\frac{7\pi}{4})$ (Fig. 8(a)).
- $\lambda(\frac{\pi}{4}, \frac{5\pi}{4})$-constraint or λ_+-constraint for brevity: Formed by the spokes of positive slope, $f_2^{k'}(\frac{\pi}{4})$ and $f_1^k(\frac{5\pi}{4})$ (Fig. 8(b)).

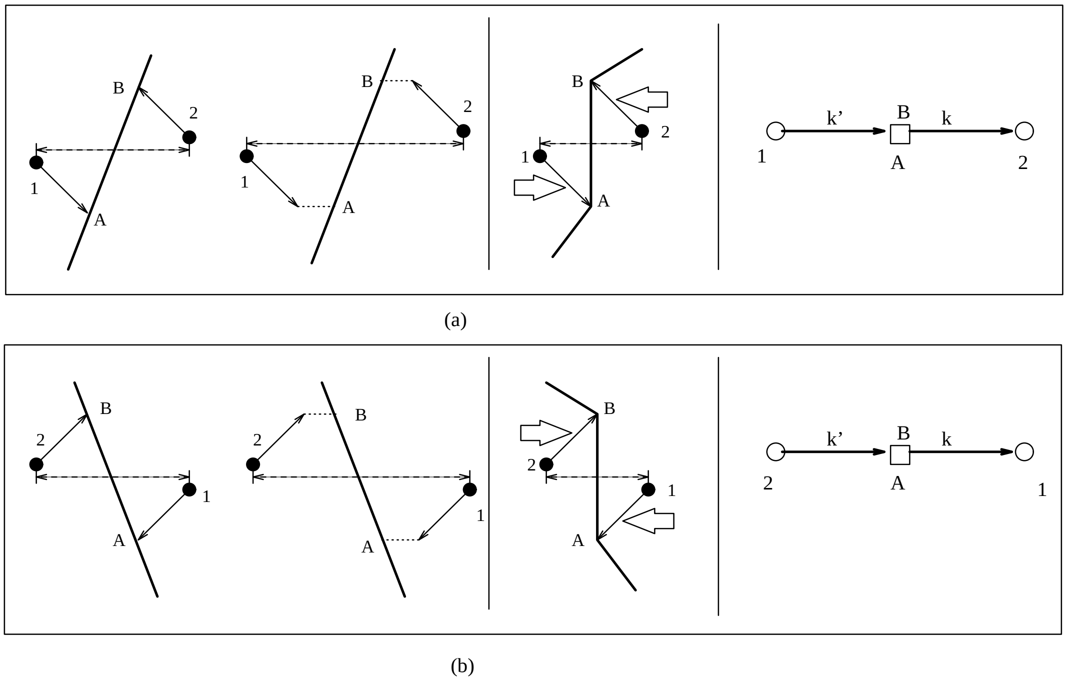


Figure 8: (a) Constraint type $\lambda(7\pi/4, 3\pi/4)$ and (b)$\lambda(\pi/4, 5\pi/4)$

Lemma 8 *Consider $\lambda(\frac{7\pi}{4}, \frac{3\pi}{4})$ and $\lambda(\frac{\pi}{4}, \frac{5\pi}{4})$ constraints between features f_1 and f_2 in a given sketch S as shown in Fig. 8. After 1-D topological compaction in x-direction, the sketch is routable if and only if the horizontal distance between features f_1 and f_2 with respect to a y-monotone chain c is at least $k + k'$, i.e.,*

$$|x(f_2) - x(f_1)| \geq k + k', \; y(A) < y(B)$$

$x(f_2) - x(f_1) \geq k + k'$ for $\lambda(\frac{3\pi}{4}, \frac{7\pi}{4})$, $x(f_1) - x(f_2) \geq k + k'$ for $\lambda(\frac{\pi}{4}, \frac{5\pi}{4})$, and A and B are the jog vertices on c induced by P-spokes f_1^k and $f_2^{k'}$ respectively.

Proof:

(only if)

Assume the compacted sketch is routable but $|x(f_1) - x(f_2)| < k + k'$. We consider the $\lambda(\frac{3\pi}{4}, \frac{7\pi}{4})$ constraint: $x(f_2) - x(f_1) < k + k'$. $\lambda(\frac{\pi}{4}, \frac{5\pi}{4})$ constraint can be proved symmetrically.

Based on the configuration, we also have $x(f_1) + k \leq x(A)$ and $x(B) + k' \leq x(f_2)$.

$$x(A) - x(B) \geq x(f_1) + k + (k' - x(f_2)) = x(f_1) - x(f_2) + k + k' > -(k + k') + k + k' = 0.$$

In other words, $x(A) > x(B)$. This implies that angles $\angle BAf_1$ and $\angle ABf_2$ are less than $\pi/4$. Thus, vertices B and A are illegal; a contradiction to the routability condition according to Lemma 3.

(if)

Since $|x(f_2) - x(f_1)| \geq k + k'$, for every pair of features f_1 and f_2, there is no illegal vertex. Since $|x(f_2) - x(f_1)| \geq k + k'$ after compaction, there is no path intersecting. Therefore, the sketch is routable.

□

Since A and B are induced by P-spokes of features f_1 and f_2, there must exist a path in the constraint graph G between the two nodes associated with features f_1 and f_2 passing through a node $\overline{AB}$, as shown in the rightmost graphs in Fig. 8. It then follows that by computing the longest path from the source to each node in the constraint graph G, the condition $|x(f_1) - x(f_2)| \geq k + k'$ must hold. The above lemma shows how the routability condition can be reflected in a general horizontal constraint graph or vice versa.

On the other hand, for jog vertices A and B defined above, where $y(A) \geq y(B)$, there is no horizontal spacing constraints between features f_1 and f_2 since no matter how the features are moved, jog vertices A and B are always legal. Thus there are two possible non-constraint types between f_1 and f_2:

- $\lambda(\frac{3\pi}{4}, \frac{7\pi}{4})$-free or λ_--free configuration for brevity: Formed by the spokes of negative slope, $f_2^{k'}(\frac{3\pi}{4})$ and $f_1^k(\frac{7\pi}{4})$ (Fig. 9(a)).
- $\lambda(\frac{\pi}{4}, \frac{5\pi}{4})$-free or λ_+-free configuration for brevity: Formed by the spokes of positive slope, $f_2^{k'}(\frac{\pi}{4})$ and $f_1^k(\frac{5\pi}{4})$ (Fig. 9(b)).

We have the following lemma.

Lemma 9 *Suppose features f_1 and f_2 form a λ_+-free or a λ_--free configuration with supporting vertices A and B as shown in Fig. 9. Then in the constraint graph G, the graph node $v \in V(G)$ associated with the y-monotone chain containing $\overline{AB}$ can be split into two nodes v_1 and v_2 such that there is no path between f_1 and f_2 in G that passes through v_1 or v_2.*

As a result of this node splitting, all the incoming edges and outgoing edges to and from v should all be partitioned accordingly. Let us assume that the y-monotone chain c gets split at vertex B, the nodes v_1 and v_2 correspond to the portions of the y-monotone chain above and below B respectively. Then those features that are *visible* from the portion of c above B will be connected to v_1 Then those features that are *visible* from the portion of c below B will be connected to v_2.

For example, as shown in Fig. 9, besides the arcs $< f_1, v_1 >$ and $< v_2, f_2 >$ in Fig. 9(a) ($< f_2, v_1 >$ and $< v_2, f_1 >$ in Fig. 9(b)), there may be other incoming arcs or outgoing arcs associated with the original node, so these arcs are also divided into two groups of incoming arcs and two groups of outgoing arcs for the two new nodes. Assume u is the node for any object to the left of the chain before splitting. The dividing rule then follows the horizontal visibility: assume the chain is split at point B, then $< u, v_1 >$ is created by any object to the left of the chain containing $\overline{AB}$ that is higher than point B ($y > y(B)$). Otherwise, arc $< u, v_2 >$ is created, since it is lower than point B ($y \leq y(B)$). If the object can see both new chains (if the object is a vertical object spanning between the vertical interval $[y_1, y_2]$, $y_1 \leq y(B) \leq y_2$), both arcs $< u, v_1 >$ and $< u, v_2 >$ are created. Similar rules apply to the nodes to the right of the chain.

In the following we discuss some issues related to the two types of spacing constraints:

False λ-Constraints

The above two conditions consider only a pair of single P-spokes in one direction, i.e., no other jog vertices between jog vertices A and B. We now consider the situation where there are other induced jog vertices between A and B. Consider Fig. 10(a) as a more general configuration on a chain where features f_i and f_j form a λ_+-constraint while features f_a and f_b form a λ_+-free configuration between jog vertices A and B induced by feature f_i and f_j. Although features a and b are shown closer to the chain in Fig. 10, $x(a)$ and $x(b)$ could be any value as long as their y-coordinates form a λ_+-free configuration such that a possible jog can be inserted. Without loss of generality, we assume the y-distance between the P-spokes from both features a and b is zero ($y(A) - y(B) = 0$) so that a jog can be inserted at point J (Fig. 10(a)). Assume

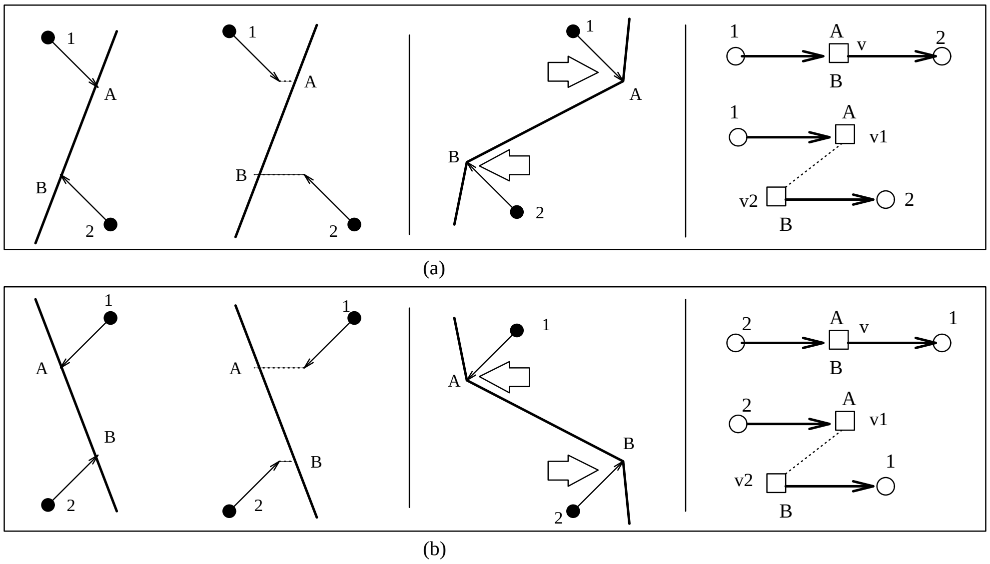


Figure 9: (a) λ_--free and (b) λ_+-free configurations

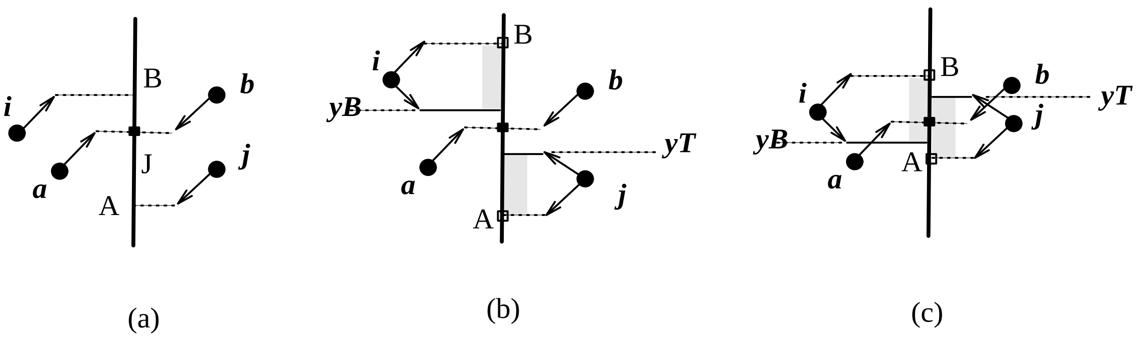


Figure 10: (a) Overlapping (b) False constraint (Case 1) (c) Real constraint (Case 2)

$y_T(j)$ is the projection of the jog vertex induced by the negative P-spoke of feature j, $y_B(i)$ is the projection of the jog vertex induced by the negative P-spoke of feature i and y_J is the location for the potential jog insertion. Depending on the configuration formed by features i and j in the -45 degree direction, there are two cases:

Case 1 : (Fig. 10(b)) features f_i and f_j form a λ_--free configuration in -45 degree direction ($y_B(i) \geq y_J \geq y_T(j)$ as shown).

Case 2 : (Fig. 10(c)) features f_i and f_j also form a λ_--constraint in -45 degree direction ($y_B(i) < y_T(j)$).

We say the λ_+-constraint formed by features f_i and f_j as shown in **Case 1** is a *false* λ_+-constraint, i.e., no constraint on the value $|x(f_i) - x(f_j)|$, because after the constraint graph is split at point J, there is no path between features f_i and f_j in the constraint graph. Otherwise, it is a real λ-constraint(**Case 2**). We have the following lemma:

Lemma 10 *Features f_i and f_j form a false λ-constraint on net $\mathcal{N}^j$ if and only if on the same net $\mathcal{N}^j$, f_i and f_j form (1) a λ_--constraint and a λ_+-free configuration, or (2) a λ_+-constraint and a λ_--free configuration.*

Note that in **Case 1** a jog at point J can be introduced without producing an illegal sketch. Therefore, the above lemma provides a criterion for jog inserion (or chain breaking) in the constraint graph. On the other hand, **Case 2** shows that certain λ-free configurations are not useful, as jogs introduced at these chain positions do not break the path connecting two features in the constraint graph that form a real λ-constraint. We say a constraint between f_i and f_j is *reducible* by a λ-free configuration, if the path between f_i and f_j in the constraint graph can be broken by the jog introduced by the λ-free configuration.

Lemma 11 *The longest path of a horizontal constraint graph G is minimal if no λ-constraints in G is reducible.*

That is, if all possible jog positions are identified by the λ-free configuration and the y-monotone chains are broken accordingly, then the compactor has the maximal flexibility to shorten all potential critical paths and still generates a routable sketch. More details can be found in [Chen 97] regarding minimizing the number of jogs that are needed to achieve the maximal flexibility.

4 The Topological Compaction Algorithm

4.1 Overview of the Algorithm

The proposed compaction algorithm combines both geometric and graph-theoretic approaches. As described, the 1-D compaction is done in two steps: the first step is to compute all the horizontal spacing constraints between the features in the layout as described in Section 2.4. The second step is to compute the critical path (longest path) in the graph and move the features (modules) accordingly without violating those constraints. In our topological approach, since wires are treated as flexible objects, jogs can be generated as long as the routability is preserved during the compaction. The routability is preserved by the constraint graph, since the constraint graph follows conditions described in Lemma 2 or Lemma 3. Based on the λ-free configurations, the constraint graph is transformed into one that contains jog information.

The first step in our algorithm is a VRBE preprocessing on the input sketch. A VRBE transformation tightens the wires so that the wire segments in the homotopy are well-behaved (i.e., no unnecessary detours).

Knowing that the jog can only occur along the y-monotone chains by transforming them to chain sketch $\overline{\overline{S}}$. This step is done in conjunction with the computation of a horizontal constraint graph G using the technique described in Section 2.4 for $\overline{S}$. That is, after applying the critical path algorithm to G, we obtain the x-coordinate value of each chain interval and the displacements of point and line features. The graph resulting from the chain sketch $\overline{\overline{S}}$, is the initial constraint graph. To cover all potential jogs that can shorten the critical path, we create the jog vertices following the λ-free configurations(Lemma 9). The collection of the jog vertices will be stored in homotopy W^1 in the new sketch S^1 (see the following algorithm). In particular, if a λ-free configuration is detected, the chain in $\overline{\overline{S}}$ will be split into two subchains and new nodes and arcs generated in the constraint graph. The new graph obtained is the decomposed constraint graph G_m associated with G. Once this graph is found, by computing its critical path, all the vertices for the features possess the information about how much they can be shifted toward the leftmost source vertex. Note that all the wires (or chains) in the chain sketch $\overline{\overline{S}}$ have already been in L_1-metric. Therefore, the new (and correct) chains provide the detailed routing. However, if the new VRBE is required, we have to collect the spokes from the features with new x-positions, and then to recover the VRBE after shifting. Note that since the compaction is performed on a VRBE sketch, only the y-monotone portion $\overline{C}$ of the homotopy will be bent. Therefore, only those portions are needed to rebuild the VRBE. To rebuild the VRBE, we need to compute the sleeves[LePr 84] from which the shortest path(the VRBE) can be obtained. When computing the jog vertices, additional information is preserved on exactly which P-spoke (of certain feature) induces the jog vertices, so the sleeves are recomputed by traversing those nets with the jog vertices after the features have been physically relocated. The VRBE can thus be computed from the sleeves following the algorithm in [ChLe96]. The following algorithm summarizes the above steps.

Algorithm 1D-TOPOLOGICAL-COMPACTION

Input : A sketch $S = (F, M, W)$.

Output: Width minimizing S^c, in which W^c is VRBE or detailed routing conforming to the original homotopy W.

Method:

1. $\overline{S}(F, M, \overline{W}) \leftarrow$ VRBE_TRANSFORMATION(S)
2. $\overline{C} \leftarrow$ CHAIN_INTERVAL($\overline{W}$)
3. $\overline{\overline{S}}(\mathbf{d}) = (F', M, \overline{\overline{C}}) \leftarrow$ CHAIN_TRANSFORMATION ($\overline{S}(F, M, \overline{C})$) :
   - $G \leftarrow$ HORIZONTAL_CONSTRAINT_GRAPH($\overline{S}$)
   - $\mathbf{d}$ = COMPUTE_CRITICAL_PATH(G) ($\mathbf{d}$= $(\vec{d_1}, \vec{d_2}, ..., \vec{d}_{|F|})$)
   - $F' \leftarrow F(\mathbf{d})$.
   - Deterimine the x-position of the chain intervals in $\overline{\overline{C}}$ based on $\mathbf{d}$.
4. $S^1(F', M, W^1) \leftarrow$ COMPUTE_JOG_VERTEX($\overline{\overline{S}}$)
5. $\overline{\overline{S}} \leftarrow$ SPLIT($\overline{\overline{S}}, W^1$)
6. $G_m \leftarrow$ DECOMPOSE($G, \overline{\overline{S}}$)
7. $\mathbf{d}_m$ = COMPUTE_CRITICAL_PATH(G_m)
8. $F'' \leftarrow F(\mathbf{d}_m)$

9. $S^c(F'', M, W^c) \leftarrow$ RECONSTRUCT_VRBE(S^1) (or output the detailed routing formed by $\overline{\overline{S}}$)
10. **return** S^c

All the implementation details except VRBE transformation are explained in the following sections. The VRBE transformation algorithm can be found in [ChLe96]. In practice, the wire homotopy W in most of the layout design after detailed routing has already been in a good shape so the VRBE preprocessing step may sometimes be saved.

4.2 Chain Transformation and Computing G

We have formulated the λ-free configuration and λ-constraint in the constraint graph between the features with the jog vertices in Section 3. In particular, if two features affect a common net (more precisely, a y-monotone chain) with their spokes, then they may have spacing constraints depending on how many wires between them in the horizontal direction and the constraint graph G can capture these spacing constraints. To compute all jog vertices over the sketch, we first need to do the chain transformation. The set of the original non-horizontal wire segments along with y-monotone chains in VRBE homotopy $\overline{C}$ is replaced by the set of the chain intervals $\overline{\overline{C}}$ in the chain sketch $\overline{\overline{S}}$. A byproduct of this step is the constraint graph G associated with $\overline{\overline{S}}$. In the following, we describe this $\overline{S}(F, M, \overline{C})$ to $\overline{\overline{S}}$ transformation (from Fig. 5 (a) to (b)).

1. Determine the set of y-monotone chains $\overline{C}$ by identifying the series of non-horizontal wire segments over $\overline{W}$.
2. Create the set of chain intervals $\overline{\overline{C}}$ by computing $I_t^j(x)$ for each y-monotone chain $\mathcal{P}_t^j \in \overline{C}$.
3. Map each feature $f \in F$ to a graph node $f \in V(G)$.
4. Map each chain interval $i \in \overline{\overline{C}}$ to a graph node $i \in V(G)$.
5. Extract visibility $E(G)$ by a plane sweep scanning over $\overline{S}(F, M, \overline{C})$.
6. Compute the longest path and determine the new x-coordinates for each $v \in V(G)$ (features and chain intervals) by topological sorting.

Steps 5. and 6. actually compact the chain sketch $\overline{\overline{S}}$. Therefore, the x-position of each chain interval $I_t^j(x)$ is defined after this transformation. Note that before the plane-sweep scan from bottom up, we have to carefully preprocess the chain representation $\overline{C}$: since the size of VRBE $\overline{W}$ could be greater than that of W in the input sketch (in the worst case, $|\overline{W}| = O(|F| \cdot |W|)$, resulting a scan spending $O((|F| + |\overline{W}|)\log(|F| + |\overline{W}|)) = O(|F| \cdot |W| \log |F| \cdot |W|)$ time). To speed up the scan, a y-monotone chain $c = \mathcal{P}_k^j \in \overline{C}$ in $\overline{S}$ is recorded with the following additional information:

1. $Start[c]$: the vertex of c with the lowest y-coordinate.
2. $End[c]$: the vertex of c with the highest y-coordinate.
3. $current[c]$: the wire segment in c intersecting the current scan line.

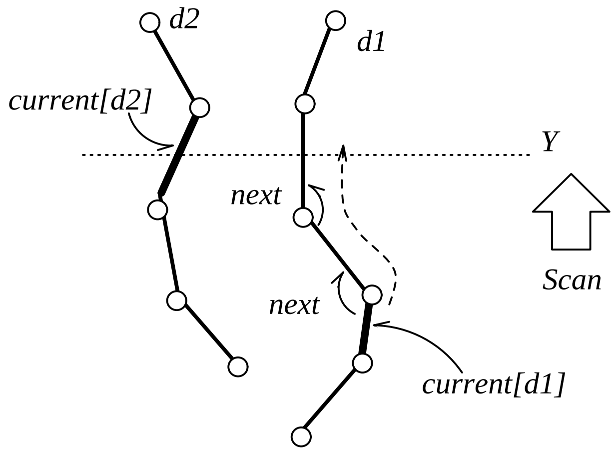


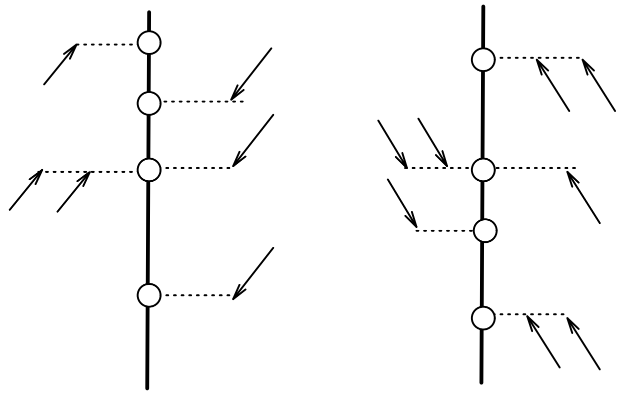
Figure 11: Pointer $current[d_1]$ for $d_1 \in \overline{C}$.

We also associate a function $currentX(c, y)$ which returns the x-coordinate of a chain c intersecting the scan line at y when it is called. Since the input is planar, no chains intersect each other. Therefore, we only need to take care of the events of starting point or ending point of chains and the features, while the rest of the events can be saved based on the planarity of VRBE $\overline{S}$. The total number of events scheduled for the scan is $|F| + 2|\overline{\overline{C}}|$ (the features and the two extremes of the chains) so that the scan can be done in $O((|F| + |\overline{\overline{C}}|)\log(|F| + |\overline{\overline{C}}|) + \mathcal{A})$ time, where $\mathcal{A}$ is the adjustment time for the joint vertices inside each chain, which will be explained next. We now describe the function $currentX(c, y)$ and value $\mathcal{A}$ (Fig. 11). During the scan, when event $Start[c]$ of chain c is encountered, $current[c]$ is initialized as the first wire segment. Recall that to check for visibility, chain c could be queried by either its left or right neighbor in horizontal direction. When $currentX(c, y)$ is called, the function will first check if $current[c]$ is the correct wire segment intersecting the current scan line. If so, it returns the corresponding x-coordinate for the given y. Otherwise, an adjustment is made to reflect the real intersecting wire segment for the corresponding y-coordinate and the correct x-coordinate is then returned. Assume the total time for the adjustment is $\mathcal{A}$. Then clearly each wire segment is only visited once by the adjustment, so $\mathcal{A} = \mathcal{O}(|\overline{\mathcal{W}}|)$ or $O(|F| \cdot |W|)$. Altogether, the scanning takes time $O((|F| + |\overline{\overline{C}}|)\log(|F| + |\overline{\overline{C}}|) + |\overline{W}| + |E(G)|)$, where $|E(G)|$ is the reporting time for the visibility which can be bounded by the size of the vertices $|V(G)|$ as described in Lemma 4. Note that $|V(G)| = |F| + |\overline{\overline{C}}| \leq |F| + |W|$ is the total number of objects in the constraint graph G. That is, the chain sketch $\overline{\overline{S}}$ and the initial constraint graph G can then be computed in $O(|F| \cdot |W|)$ time.

Lemma 12 *The horizontal constraint graph $G(V, E)$ associated with a chain sketch $\overline{\overline{S}}$ derived from VRBE $\overline{S}$ of N_1 objects ($N_1 = |F| + |\overline{\overline{C}}| = |F| + |\overline{C}| \leq |F| + |W| = N$) can be computed in time $O(N_1 \log N_1 + |\overline{W}| + |E(G)|)$, where $\overline{W}$ denotes the set of wires in VRBE $\overline{S}$. Moreover, $|V(G)| = N_1$, $|E(G)| = O(N_1)$, and the longest path can be computed by topological sorting in time $O(|F| + |W|)$.*

4.3 Computing Jog Vertices

In this section, we describe the procedure of locating the jog vertices over the chain intervals in $\overline{\overline{C}}$. For each feature f, as defined in Section 3, there are four possible directions so there are four types of spokes:



(a) (b)

Figure 12: The set W^1 of ordered lists: (a) $\mathcal{L}^+(c)$ (b) $\mathcal{L}^-(c)$

$f^k(\theta)$, $\theta = \alpha\pi/4, \alpha = 1,3,5,7$. Moreover, among the spokes $f^k(\theta)$, two of them are *upward* ($\pi/4$ and $3\pi/4$) and two of them are *downward* ($5\pi/4$ and $7\pi/4$). We show that the *upward* spokes can be extracted by a plane-sweep scan. The *downward* spokes can be extracted by a top-down scan symmetrically.

In the following, given $\overline{\overline{C}} = \overline{\overline{W}}$ from chain sketch $\overline{\overline{S}}$, we describe the bottom-up scan computing jog vertices induced by upward spokes. Assume $c \in \overline{\overline{C}}$ and $c = I_t^j(x)$ is a chain interval. The identified jog vertices associated with each chain interval $c \in \overline{\overline{C}}$ is stored as an ordered list $\mathcal{L}(c)$ (ordered in y-coordinate). The set W^1 collects all these ordered jog vertex lists:

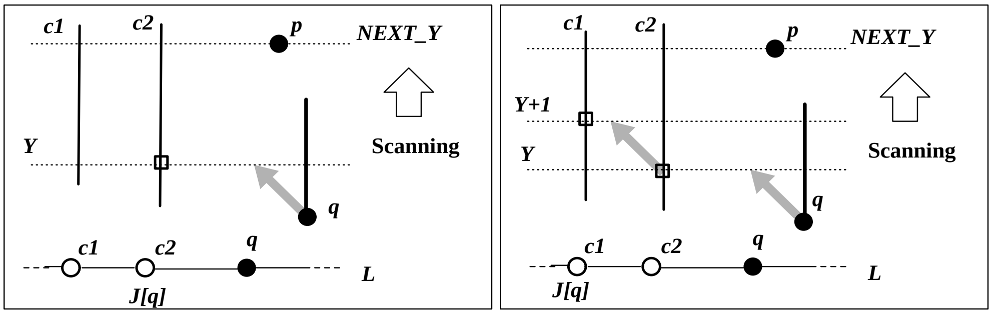
$$W^1 = \bigcup_{c \in \overline{\overline{C}}} \mathcal{L}(c)$$

Since we have two types of constraints: λ_+ and λ_-, the jog vertices can be divided into two groups: one group is solely determined by the negative P-spokes in $3\pi/4$ and $7\pi/4$ directions while the other is by positive P-spokes in $\pi/4$ and $5\pi/4$ directions. Therefore, we associate lists $\mathcal{L}^+(c)$ and $\mathcal{L}^-(c)$ with c, such that $\mathcal{L}^+(c)$ is the list (ordered in y-coordinate) of jog vertices induced by positive P-spokes on c, while $\mathcal{L}^-(c)$ for jog vertices induced by negative P-spokes on c. That is, W^1 in the resulting S^1 will be $W^1_+ \cup W^1_-$, such that (Fig. 12)

$$W^1_+ = \bigcup_{c \in \overline{\overline{C}}} \mathcal{L}^+(c)$$

$$W^1_- = \bigcup_{c \in \overline{\overline{C}}} \mathcal{L}^-(c)$$

Initially $\mathcal{L}^+(c)$ and $\mathcal{L}^-(c)$ are empty. In a bottom-up scan, we only consider the upward P-spokes in directions $\pi/4$ and $3\pi/4$. During the bottom-up scan, an upward P-spoke (i.e., in $\pi/4$ and $3\pi/4$ directions) can not be defined until the scan line passes over its feature. Whenever a $\pi/4$ direction P-spoke is identified, the associated jog vertex induced on chain interval c will be inserted into list $\mathcal{L}^+(c)$, while a $3\pi/4$ one will be inserted into $\mathcal{L}^-(c)$. Note that in a bottom-up scan, the jog vertices are detected sequentially from bottom to top along a chain interval. Therefore, the action of adding jog vertices to the $\mathcal{L}$ lists behaves like a push-down stack operation (an enqueue operation for top-down scan). Only one jog vertex is needed in case there are more than one spoke projected onto c at the same y-coordinate but the P-spokes should all be recorded.



(a) (b)

Figure 13: Spoke expansion

For each event point, two tasks are performed: *(1) Event status updating* and *(2) spoke expansion*. The first task is the standard operation to either insert or delete the chain intervals in $\overline{\overline{C}}$, feature segment or feature point in F. In the following, let the status be list L in which the horizontal order of the objects are preserved so that in the list, finding the immediate neighbor either to the left or to the right for any node takes $O(1)$ time. The second task, spoke expansion, involves finding the appropriate jog vertex described as in Definition 2 for each of the P-spokes of the features. We use another list Γ to record the features with either $3\pi/4$ or $\pi/4$ upward spoke encountered so far. Define $J[q]$ as a pointer to the elements in list L, $J[q] = c \in L$, for some feature $q \in \Gamma$. A jog vertex at the current y-coordinate will be added to $\mathcal{L}$ list of the chain interval pointed by $J[q]$. This expansion task can be explained by Fig. 13.

In this figure, points p and q are two features and c_1 and c_2 are two chain intervals already in the status list L. Note that q is one of the features just passed by the (bottom-up) scan. Meanwhile, $Y = y[q] + 1$ is the current y-coordinate under consideration. Since q is below Y as shown in the figure, the spoke associated with q is expanded and $J[q]$ points to node (c_2) immediately to the left of feature q. The $3\pi/4$ P-spoke of rank one associated with q is shown in Fig. 13(a). At this point, a jog vertex is created at Y on c_2. Because Y is lower than the next event point p, the next P-spoke of rank two associated with q, $q^2(3\pi/4)$, is expanded next by moving Y up one unit and updating $J[q]$ to be the next chain interval, c_1, as shown in Fig. 13(b). Note that each feature involves in at most $|\overline{\overline{C}}|$ (or $O(|W|)$) expansions (Lemma 5). In particular, when $J[q]$ points to a vertical line feature, the spoke associated with q will be removed from list Γ. This situation is illustrated in Fig. 7 where the P-spoke propagation stops when it hits a rigid object other than a chain. Besides $J[q]$, we also associate additional attributes to node for spoke q in list Γ: $f[q]$ for the feature, $k[q]$ for the rank, and $\theta[q]$ for the orientation of the spoke inducing a jog vertex. This information is essential for recomputing VRBE in the last step of the compaction.

The following algorithm details how the events and scanning status are maintained, where steps 11 to 23 perform the first task (event status updating) for all objects at the same y-coordinate and steps 24 to 43 for the second task (spoke expansion).

Algorithm COMPUTE_JOG_VERTEX

Input : Chain sketch $\overline{\overline{S}}(F, M, \overline{\overline{C}})$

Output: $S^1(F, M, W^1)$ such that the wire homotopy includes jog vertices

Method:

```
for every  extreme or endpoint p of the wires and features in S.
    do
        Q ← Enqueue(Q, p)
    od
Y ← Min_y[S]
Γ ← nil
L ← nil
while Q not empty
        do
            p ← Dequeue(Q)
            while Y = y[p]
                    do /* collect all objects at the same y-coordinate. */
                        case 1 : p is an end-point of a chain interval c
                                    Update L by either Insert(L, Start[c]) or
                                    Delete(L, End[c])
                        case 2 : (1) p is an end-point associated with a feature f_i
                                    if f_i is associated with a vertical "line feature" l.
                                    J[p] =Insert(L, l(f_i)) or Delete(L, l(f_i))
                                    (2) f[p] ← f_i    k[p] ← 0
                                    either   θ[p] ← 3π/4 or   θ[p] ← π/4
                                    depends on which side of a module f_i belongs to.
                                    If f_i is on top side of a module or an isolated
                                    point, both instances are inserted into Γ
                                    Γ ← Insert(Γ, p)
                        p ← Dequeue(Q)
                    od
            Next_Y = y[p]
            while Y ≤ Next_Y
                    do
                        for every spoke q in Γ associated with f_i(θ)
                            do
                                if k[q] ≥ 1
                                  then
                                  J[q] ← left[J[q]],   if θ[q] = 3π/4
                                  J[q] ← right[J[q]],   if θ[q] = π/4
                                  if J[q] is a feature but not a local minimum
                                    then
                                          /* spoke expansion for q stops */
                                          Delete(Γ, q)
                                    else
                                         c ← J[q]
                                         Add-Jog-Vx(Y, c, q)
                                  fi
```

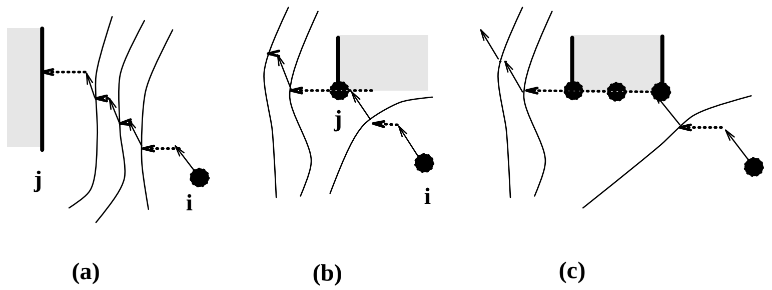


Figure 14: Physical blocking and pseudo blocking.

fi
$k[q] \leftarrow k[q] + 1$
od
$Y \leftarrow Y + 1$
od
$Y =$ NEXT_Y
od

We highlight some points in the following: to implement the scanning in this algorithm, we have a priority queue Q and the associated operations ENQUEUE() and DEQUEUE(), where the latter takes the event point with the smallest y-coordinate in Q. L and Γ are realized by any height-balanced tree with updating operations INSERT() and DELETE() in time $O(\log(|F| + |\overline{\overline{C}}|))$ (the total number of scanning events is $|F| + 2|\overline{\overline{C}}|$). In L, $right[q]$ and $left[q]$ can access the data node next to data node q immediately. We now explain how function ADD-JOG-VX() should insert the appropriate jog vertex into either $\mathcal{L}^-(c)$ or $\mathcal{L}^+(c)$. Note that in a horizontal compaction, only x-coordinates of the objects are displaced, so for each jog vertex, its y-coordinate is well-defined according to the scan, and the x-coordinate of the final VRBE can be computed from the relative x-coordinate of the features that induced the P-spokes. For the purpose of recomputing the VRBE for $\mathcal{P}_k^j$, for each jog vertex v pushed to $\mathcal{L}$ lists, we associate the following attributes tagged by function ADD-JOG-VX().

- $y[v]$: The position of the horizontal projection on chain interval c from P-spoke $f[q]^{k[q]}(\theta[q])$ of q in list Γ. Note that, during the subsequent 1-D compaction, $x[v]$ will be determined.
- $feature[v]$, $dir[v]$ and $rank[v]$: Record the information $f[q]$, $\theta[q]$ and $k[q]$ about the spoke potentially supporting it in the future. These attributes provide sufficient information to recover the VRBE wire homotopy after compaction.
- $pre[v]$ and $next[v]$: Record the previous and next jog vertices along the chain.

To complete the jog vertices lists, after the bottom-up scan we have to perform a top-down scan. The final $\mathcal{L}^+(c)$'s are obtained by merging the lists consisting of jog vertices from $\pi/4$ spokes (produced by bottom-up scan) and lists consisting of jog vertices from $5\pi/4$ spokes (produced by top-down scan). The final $\mathcal{L}^-(c)$'s are obtained similarly by merging the lists formed by $3\pi/4$ and $7\pi/4$ spokes. After the above scannings and mergings steps, all ordered lists $\mathcal{L}^+(c)$ and $\mathcal{L}^-(c)$ are computed.

We have a comment on step 32 in the algorithm : In Fig. 14(a), if the next object f_j that the spoke $f_i^k(\theta)$ touches is either a vertical line feature or a local maximum, it means there should be a spacing constraint reported between features i and j in the yet to be constructed constraint graph. Note that in the expansion steps, feature j could be one of the leftmost or rightmost pseudo feature **L** or **R** mentioned earlier. On the other hand, if the next feature object f_j is a local minimum like the configuration shown in Fig. 14(b), it implies that the current spoke may still have the chance to *see through* the local minimum a chain interval. For instance, after compaction, feature f_i may be shifted left passing over f_j. Therefore, the spoke expansion should continue. In fact, the case shown in Fig. 14(c) could also occur (with multiple local minimum features in a row). Those local minimum features in a group should simply be ignored during expansion. To implement this "see through" ability, before step 32, the following loop is needed to skip all the local minima as in case (c):

while $J[q]$ is a local minimum
continue

On the other hand, for a top-down scan, the condition in step 32 should change to "if it is not a local maximum then the expansion stops..".

Lemma 13 *Given a chain sketch $\overline{\overline{S}}(F, M, \overline{\overline{C}})$, W^1, the jog vertex lists, can be computed in time $O(N_1 \log N_1 + |J|)$, where $N_1 = |F| + |\overline{\overline{C}}|$ is the size of $\overline{\overline{S}}$ and $|J|$ is the total number of jog vertices which is $O(|F| \cdot |W|)$.*

That is, this step takes time $O(|F| \cdot |W|)$.

4.4 Computing G_m

After jog vertex expansion, we have obtained new homotopy representation W^1 in which for each chain interval $c \in \overline{\overline{C}}$, there are two ordered lists $\mathcal{L}^+(c)$ and $\mathcal{L}^-(c)$ consisting of jog vertices induced by positive and negative P-spokes, respectively. Note that W^1 representation enables us to locate the potential jogs. That is, following the λ-free configurations, we can traverse W^1 to split the chain intervals into pieces forming jogs. This means more nodes are introduced into the constraint graph and the new graph node set is determined by the nodes induced by the split chains. On the other hand, the corresponding visibility changes in $E(G)$ should also reflect the new visibility between the nodes in $V(G_m)$ accordingly. As defined, the new graph resulting from the split new chain intervals is the decomposed constraint graph G_m, so the split chains induce new graph nodes forming $V(G_m)$ while new visibility determines $E(G_m)$. Since the chains do not intersect and the constraints between adjacent chains (after jog insertion) can be generated in time $O(1)$ per chain segment by a merge operation, the size of graph G_m is no larger than the size of the possible number of jogs, or in turn, $O(|F| \cdot |W|)$:

Lemma 14 *After chain splitting, graph G_m can be derived from G in time $O((|F|+|\overline{\overline{C}}|)\cdot|F|)$ or $O(|F|\cdot|W|)$ when $|W| > |F|$, $|V(G_m)| = |F| + |F| \cdot |\overline{\overline{C}}| = O(|F| \cdot |W|)$ and $|E(G_m)| = O(|E(G)| \cdot |F|) = O(|F| \cdot |W|)$.*

Since the graph size is $O(|F| \cdot |W|)$, we conclude that

Lemma 15 *The final compacted width of a sketch $S(F, M, W)$ can be computed in time $O(|V(G_m)| + |E(G_m)|)$, or $O(|F| \cdot |W|)$.*

Note that after the longest path is obtained, every feature and chain also receive their correct x-position after compaction.

4.5 Wire Homotopy Reconstruction

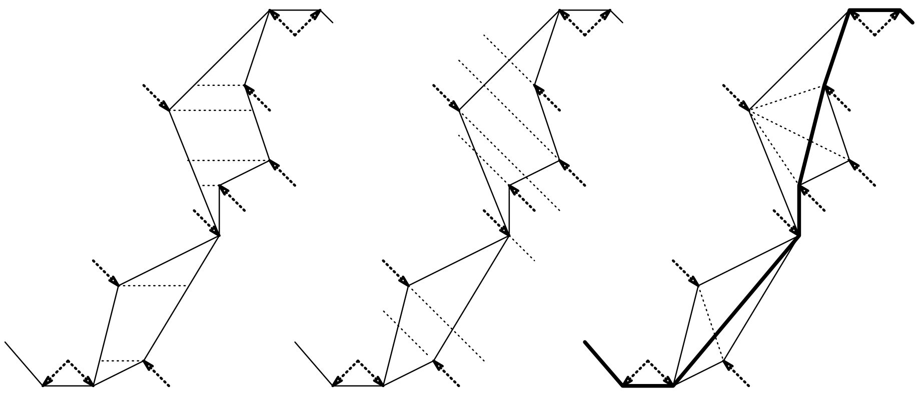
After computing the longest path in G_m, all the features and the chain intervals (vertical line segments $I_t^j(x)$) have received their correct x-positions (y-positions are the same as the y-monotone chains before compaction). In this section, we consider how the wire homotopy can be reconstructed in the compacted sketch with the new x-positions. If the given sketch S is already detailed routed in L_1 metric, then all we need to do is to connect those new (vertical) chain intervals with horizontal wire segments (the shaded horizontal portions in Fig. 5(b)). Otherwise, if the VRBE is the required homotopy representation, it can be recovered by the information associated with each jog vertex. The information has been recorded during spoke expansion. Since each jog vertex carries the information about the spoke affecting it, the distance and the direction, the new positions of the jog vertices determine the envelope of the sleeve corresponding to the new VRBE subnet for the original y-monotone chains in $\overline{C}$. Note also that only x-coordinates of the jog vertices are changed after computing G_m and they can be translated relative to the features. For example, the new x position of a jog vertex v in certain chain lists $\mathcal{L}^+(c)$ and $\mathcal{L}^-(c)$ can be determined by

$$x[v] = x[feature[v]] - rank[v],\ if\ dir[v] = 3\pi/4\ or\ 5\pi/4$$

or

$$x[v] = x[feature[v]] + rank[v],\ if\ dir[v] = \pi/4\ or\ 7\pi/4$$

with $y[v]$ unchanged. To recompute the final vertex-disjoint RBE W^c after compaction, we adopt a similar decomposition approach used in [ChLe96][LeMa 85]: Let W^c_- and W^c_+ be the shortest paths determined by the sleeve formed by spokes in $\{3\pi/4, 7\pi/4\}$ direction and $\{\pi/4, 5\pi/4\}$ direction, respectively. Consider $\mathcal{L}(\hat{\mu}^j_{t-}) = \mathbf{d}_m(\mathcal{L}^-(c))$, $\hat{\mu}^j_{t-} \in W^c_-$, the list of the jog vertices in $\mathcal{L}^-(c) \in W^1_-$ after compaction where all the features are translated with the vector $\mathbf{d}_m = (d_1, d_2, \ldots, d_{|F|})$ as shown in Fig. 15(a). From the jog



(a) (b) (c)

Figure 15: Sleeve retrieving and RBE computing.

vertices of $\mathbf{d}_m(\mathcal{L}^-(c))$, we can collect the spokes from both sides into two lists that serve as the boundaries (the envelope) of the new sleeve. We then perform a merging according to the -45 degree projections of the spoke tips (i.e., the jog vertices in new positions) as shown in Fig. 15(b))to collect the points in an ordered list which is exactly the new sleeve itself. A natural triangulation embedding is provided after the merging and the shortest path can be computed using the procedure proposed in [ChLe96][LePr 84] (Fig. 15(c)). The shortest path in $\mathcal{L}(\hat{\mu}^j_{t+})$ can be computed similarly. The new VRBE $\hat{\mathcal{N}}^j \in W^c$ can thus be composed from the $\mathcal{L}(\hat{\mu}^j_{t+})$ and $\mathcal{L}(\hat{\mu}^j_{t-})$; the details can be found in [ChLe96][LeMa 85]. With additional transformations to VRBE, the detail routing methods proposed in [Gao 89] can also be applied to reconstruct the wire homotopy.

5 Complexity Analysis

The VRBE preprocessing takes time $O(|F| \cdot |W|)$ (Lemma 1). To construct the horizontal constraint graph G, the algorithm visits each of the jog vertices so its time complexity is $O(N \log N + |\overline{W}|) = O(|F| \cdot |W|)$ (Lemma 12, $N = |F| + |W|$). The step COMPUTE_JOG_VERTEX() takes time $O(N_1 \log N_1 + |F| \cdot |W|)$ for both scanning and reporting for all the jog vertices (Lemma 13, where $N_1 = O(|F| + |W|)$). The time for finding the critical path is also proportional to the size of the acyclic digraph. Since the number of arcs may be at most the number of the jog vertices, the time complexity again is $O(|F| \cdot |W|)$ (Lemma 15). Finally, shifting the features and wires takes $O(|F| + |F| \cdot |W|)$. Therefore, the overall running time of the compaction algorithm is $O(|F| \cdot |W|)$. Based on the lower bound argument for a sketch model given in [LeMa 85], which is $\Omega(|F| \cdot |W|)$, the overall time complexity is asymptotically optimal in the worst case. This complete the proof of **Theorem 1**.

6 Extensions

Mulit-Terminal Nets Sketch Topological Compaction Problem

A transformation from a sketch with multi-terminal nets to two-terminal nets proposed by Kaufmann *et al.*[KaGa 94] was adopted as a preprocessing for VRBE on multi-terminal nets. The same transformation can also be used for the multi-terminal sketch topological compaction problem. Basically for each s-terminal ($s > 2$) net, the preprocessing decomposes it into s two-terminal nets on a finer grid plane. The resulting sketch then can serve as the input of the compaction algorithm. After compaction, a Steiner minimal tree algorithm [KaGa 94] is then applied to recover the best topology of the original multi-terminal nets in time $O(s^2 \cdot \min\{s \log s, v\})$, where v is the total number of grid points inside the rectilinear polygon enclosed by the multi-terminal nets. Therefore, the time bound should be that for the compaction algorithm itself plus the time used to recover the Steiner minimal tree for the multi-terminal nets.

Multi-Layer Sketch Topological Compaction Problem

Nowadays multi-layer VLSI layouts with layer number more than three is very common. However, there is no method reported to date to introduce jogs systematically in a multi-layer layout topological compaction problem. In multi-layer layout a set of non-crossing nets may be mapped to a preferred layer and routed in a *river routing*[CoSi 84] (single-layer routing). In this case there is no *via* or *contact* in the same net, so it is a direct application using the proposed algorithm to compute the spacing constraints in different layers simultaneously. Once all the features are determined, VRBE can be recovered in each layer. For multi-layer VLSI layout with nets containing via, the related issue is to determine the positions of the *crossings* (the intersection between nets). In [Groe 89] [MaSa 95] [ChLe98], *crossing minimization problem* and *crossing distribution problem* are considered in the given global routing topology as a preprocessing. The unavoidable crossings between the nets are placed over the layout regions in a way such that the best area utilization and minimal number of tracks used are attained for detailed routing later. Therefore, before applying the proposed algorithm to a multi-layer layout compaction problem, such algorithms may be used as preprocessing over the layout so that the crossings are fixed within some regions. We then add vias around the crossings and treat the vias as new features in the sketch. The vias can be introduced using the available *constrained via minimization* algorithm in [SL 89] on the given sketch such that the total number of vias introduced is minimal. Let ξ be the number of necessary crossings in the sketch. We introduce at most $O(\xi)$ vias. Additional $O(\xi)$ wire segments may also be formed. In summary, with the preprocessings of

1. crossing minimization
2. crossing distribution
3. constrained via minimization

we may have a set of related sketchs $\overline{S_\ell}$, where ℓ is the total number of layers available. Assume each crossing in ξ results in an additional wire segment in W and two more vias in F. Since the size of feature set F and wire set W become $|F| + 2\xi$ and $|W| + \xi$ respectively, the running time for multi-layer sketch topological compaction should be $O(\ell \cdot ((|F| + \xi) \cdot (|W| + \xi))$.

Minimizing Total Wire Length

There are arguments on the disadvantages resulting from jog insertion that too many jogs imply the nets may have many detours though the minimum width is achieved. That is, the objectives of minimizing the overall width and minimizing the total wire length sometimes may conflict with each other: in order to satisfy the minimal width requirement, some nets must be longer because of jog insertion. Note that when computing the critical path in the constraint graph, each node keeps the minmal value from left (x-coordinate). This implies that the wire will be shifted leftward as much as possible. This may not be necessary, especially when there is room for moving rightward (positive x coordinates) without increasing the total width. This *rightward movable room* can be computed from a reverse acyclic graph constructed from G_m such that the directions of all the arcs are reversed and the maximal values are computed from right to left. Thus, each node has a Δx value representing the range between the leftmost x position and the rightmost x position that the node can be placed. In our algorithm the trade-off can be accomplished by controlling the chain intervals so that some features or modules are not shifted unnecessarily toward left and detoured too much. Moreover, in the extreme case, when we straighten the chains without any jog, the wire length is minimal (no detours at all). Recall that in our algorithm, a y-monotone chain is recovered by retrieving its sleeve. We propose the following greedy algorithm. The idea is to *straighten* the monotone nets whenever it is possible while the overall width is not increased based on the Δx. The algorithm may start from the sleeve of either

1. a critical net,
2. a net with maximal length, or
3. the rightmost y-monotone chain.

and try to push the features inducing jog vertices on the net to the right direction to minimize the length.

7 Conclusion

In this paper we have presented an efficient and simpler single-layer 1-D topological compaction algorithm that improves over previous results [Gao 89] [MeNa 90] [Male 91] on topological compaction problem. Although the proposed algorithm is only for a planar routing model, it can be extended to multi-layer technology. It generalizes easily to applications like cell-level layout, macro-level, building block layout, or general-architecture layout etc. Since the algorithm also generates a horizontal constraint graph, plowing operation can be added to improve the layout quality.

The novel part in our topological compaction is that we solve the compaction problem with jog insertion using an efficient graph algorithm combined with geometry that helps identify all potential jog vertices so that the width can be minimized. Note that previous topological compactors have costly operations such as maintaining the density or capacity information or updating the critical cuts by monitoring all the possible triangles among the features. Furthermore traditional graph-based compactors cannot solve the problem exactly either.

The algorithm was implemented in C on a geometric input/output device, called GeoSheet, which is part of a geometric object manipulation and monitoring system developed at the Northwestern University. Fig. 16 and Fig. 17 show the computer output of two test examples in which diagram (a) gives the input sketch, (b) gives the result after conventional 1-D compaction without modules, (c) gives the result after 1-D compaction with modules, (In fact, (b) and (c) are the chain sketch $\overline{\overline{S}}$ corresponding to the initial sketch S in (a).) (d) gives the result after the proposed topological compaction without modules, and (e) gives the result after topological compaction with modules. In Fig. 18, we show another larger example with more modules and nets and its output. Table 1 shows CPU time (SPARC 10) of some test examples of various input sizes, where $\mathcal{W}_0$ represents the width of the sketch after conventional 1-D compaction (i.e., Fig. 16(c)) and $\mathcal{W}_1$ represents the width of the sketch after applying our 1-d topological compaction (i.e. Fig. 16(e)). It seems to indicate that the running time is roughly proportional to the product of $|F|$ and $|W|$ (Fig. 19).

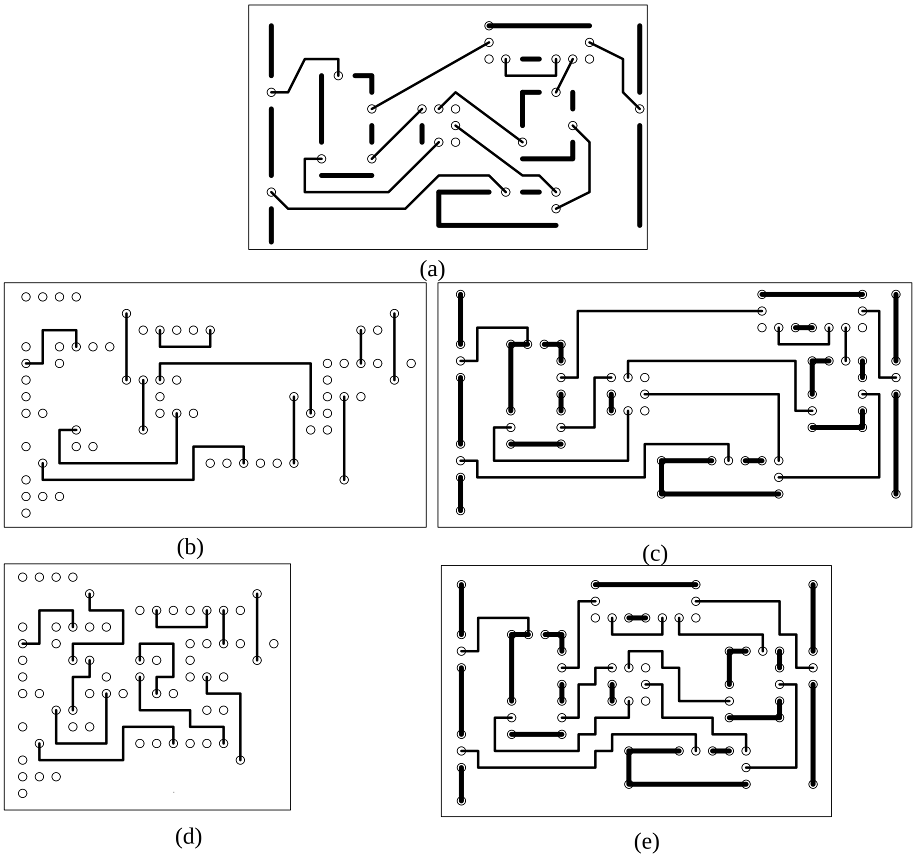


Figure 16: Test example S_1.

References

[Card 91] R.C. Carden IV and C. K. Cheng, "A Global Router using an efficient Approximate Multi-commodity Multiterminal Flow Algorithm", *Proc. 28th Design Automation Conf*, pp.316-321, 1991.

[Chen 97] H.-F. S. Chen, "Topological Optimization for VLSI Layout Design", *Ph.D. Thesis, Northwestern Univeristy*,Evanston, IL, June 1997.

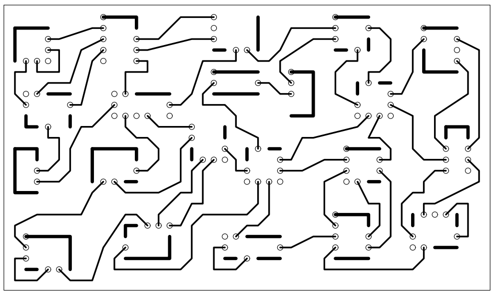
[ChLe96] H. F. S. Chen and D. T. Lee, "A Faster Algorithm for Rubber-Band Equivalent Transformation for Planar VLSI Layouts", *IEEE Trans. Computer-Aided Design*, vol.15, Feb. 1996, pp. 217-227.

[ChLe98] H. F. S. Chen and D. T. Lee, "On Crossing Minimization Problem," *IEEE Trans. Computer-Aided Design*, vol. 17, no. 5, May 1998, pp. 406-418.

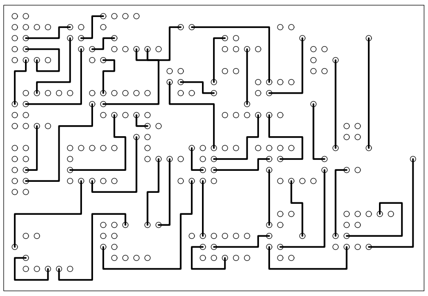
[ChSa 90] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "Global Routing Based on Steiner Minmax Trees", *IEEE Trans. Computer-Aided Design* vol. CAD-9, no. 12, pp. 1318-1325, Dec. 1990.

[CoSi 84] R. Cole and A. Siegel, "River Routing Every Which Way, but Loose," in *Proc. 25th Ann. Symp. on Foundations of Computer Science*, Oct. 1984, pp.65-73.

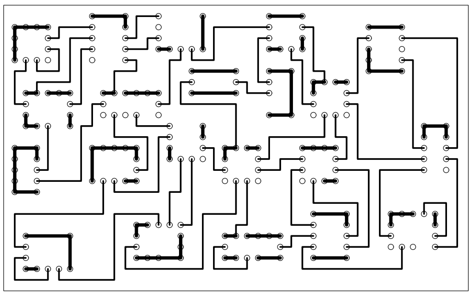
[DaKo 90] W. W.-M Dai, R. Kong, and J. Jue, "Rubber Band Routing and Dynamic Data Representation", in Proc. ICCAD, 1990, pp.52-55.



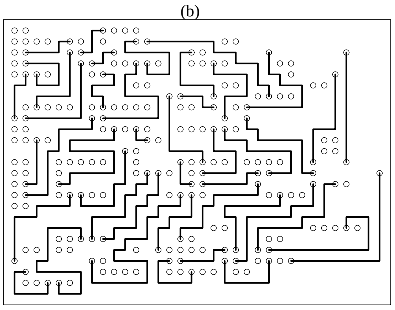
(a)



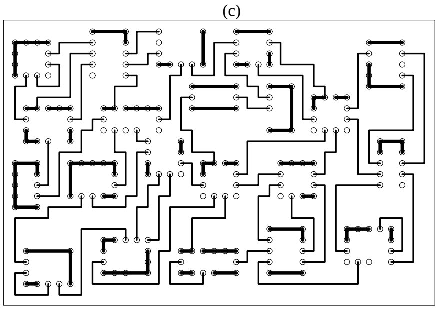
(b)



(c)

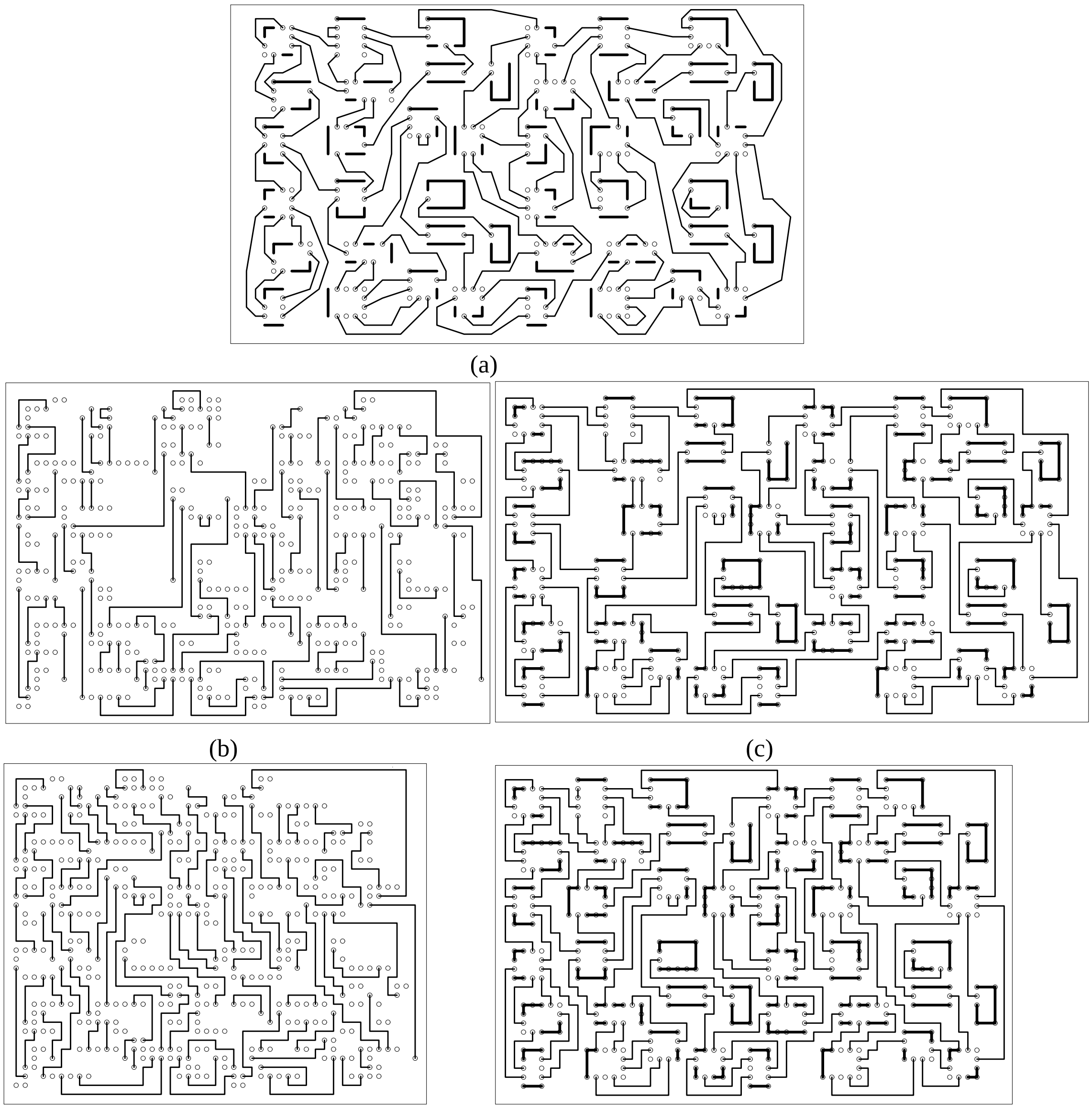


(d)



(e)

Figure 17: Test example S_6.



(a)

(b)

(c)

(d)

(e)

Figure 18: Test example S_{10}.

sketch	S_1	S_2	S_3	S_4	S_5
$\|F\|$	65	113	113	225	225
$\|W\|$	41	105	141	204	241
$\|M\|$	5	11	11	21	21
# of nets	11	21	26	41	44
$\|F\| \cdot \|W\|$	2665	11865	15933	45900	54225
$\mathcal{W}_0$	260	300	280	370	410
$\mathcal{W}_1$	210	260	260	350	380
$\frac{\mathcal{W}_0 - \mathcal{W}_1}{\mathcal{W}_0}\%$	19.2%	13.3%	7%	5%	7%
cpu time (sec.)	0.2	0.4	0.5	1.5	1.8
sketch	S_6	S_7	S_8	S_9	S_{10}
$\|F\|$	307	307	378	378	452
$\|W\|$	199	315	218	391	446
$\|M\|$	28	28	34	34	44
# of nets	44	61	42	75	88
$\|F\| \cdot \|W\|$	61093	96705	82404	147798	201592
$\mathcal{W}_0$	430	440	590	600	630
$\mathcal{W}_1$	400	440	470	510	550
$\frac{\mathcal{W}_0 - \mathcal{W}_1}{\mathcal{W}_0}\%$	6%	0%	20%	15%	12%
cpu time (sec.)	2.1	3.3	3.7	6.4	9.9

Table 1: Topological Compaction experimental results

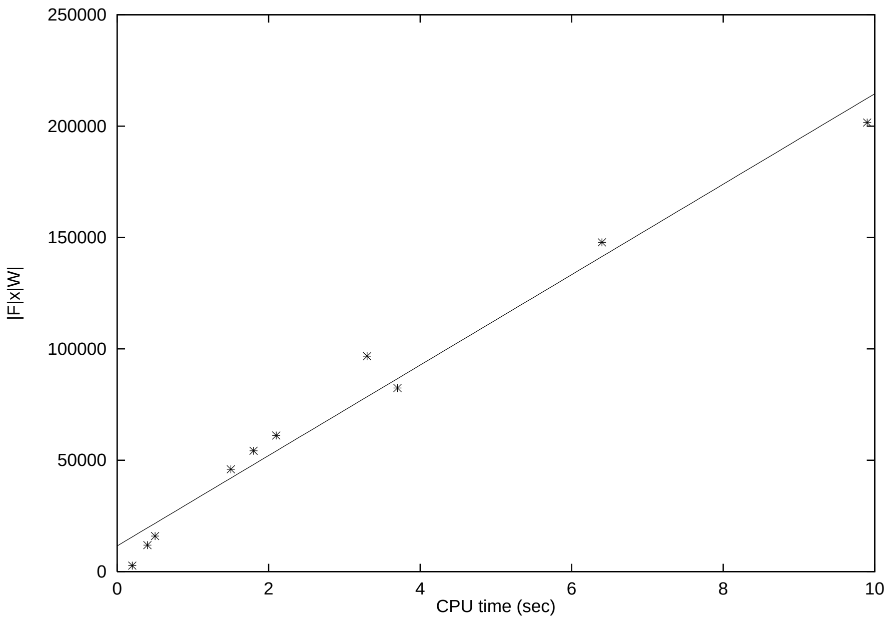


Figure 19: Correlation

[DaDa 91] W. W.-M. Dai, T. Dayan and D. Staepelaere, "Topological Routing in SURF: Generating a Rubber-Band Sketch", *Proc. of the 28th ACM/IEEE Design Automation Conference*, pp.39-44, 1991.

[Dun 78] A. E. Dunlop,"SLIP: Symbolic Layout of Integrated Circuits with Compaction", *IEEE Trans. Computer-Aided Design*, Vol.10, Nov. 1978.

[Gao 88] S. Gao, M. Jerrum, M. Kaufmann, K. Mehlhorn, W. Rülling, and C. Storb,"On Continuous Homotopic One Layer Routing", *Proc. of the 4th Annual Symposium on Computational Geometry, ACM*, 1988, pp. 392-402.

[Gao 89] S. Gao, M. Kaufmann, and F. M. Maley, "Advances in Homotopic Layout Compaction", *Proceedings of the 1st ACM Symposium on Parallel Algorithms and Architectures*, pp. 273-282, 1989.

[Groe 89] P. Groenveld, "On Global Wire Ordering for Macro-cell Routing," *Proc. Design Automation Conference*, 1989, pp. 155-160.

[HSM 93] J.-M. Ho, A. Suzuki, and M. Sarrafzadeh, "An Exact Algorithm for Single-Layer Wire Length Minimization", *IEEE Trans. on Computer-Aided Design*, pp. 175-180, Jan., 1993.

[Kao 95] W.-C. Kao, and T.-M. Parng, "Cross Point Assignment with Global Rerouting for General-Architecture Designs", *IEEE Trans. Computer-Aided Design.* vol. CAD-14, No.3, pp.337-348, March, 1995.

[KaGa 94] M. Kaufmann, S. Gao, and K. Thulasiraman, "On Stenier Minimal Trees in Grid Graphs and Its Application to VLSI Routing," in *Proc. 5th Int. Symp. Algorithms and Computation*, Beijing, China, pp. 351-359, Aug., 1994.

[KaMe 94] M. Kaufmann and K. Mehlhorn, "A Linear Time Homotopic Routing Algorithm", *SIAM on Computing*, pp. 227-246, vol. 23, No.2, April, 1994.

[KeWa 84] G. Kedem and H. Watanbe, "Graph-Optimization Techniques for IC-layout and Compaction", *IEEE Trans. Computer-Aided Design*, Vol.3, pp.12-20, 1984.

[LePr 84] D. T. Lee and F. P. Preparata, "Euclidean Shortest Paths in the Presence of Rectilinear Barriers", *Networks*, Vol. 14, pp.393-410, 1984.

[LeMa 85] C. E. Leiserson and F. M. Maley, "Algorithms for Routing and Testing Routability of Planar VLSI Layouts," *in Proc. 17th Annu. ACM Symp. on Theory of Computing*, 1985, pp.69-78.

[Leng 90] T. Lengauer, "Combinatorial Algorithms for Integrated Circuit Layout", John Wiley & Sons, 1990.

[LiMa 84] J. T. Li and M. Marek-Sadowski, "Global Routing for Gatearray", *IEEE Trans. Computer-Aided Design.* vol. CAD-4, pp.298-307, Oct., 1984.

[LiSa 91] K.F. Liao and M. Sarrafzadeh. "Boundary Single Layer Routing with Movable Terminals". *IEEE Trans. on Computer- Aided Design of Integrated Circuits*, Vol.10, pp.1382-1391, Nov. 1991.

[LoVa 90] C.-Y. Lo, and R. Varadarajan,"An $O(n^{1.5} \log n)$ 1-D Compaction Algorithm ", *Proc. of the 26th Design Automation Conf.*, pp.382-387, 1990.

[Male 91] F. M. Maley, "A Generic Algorithm for One- Dimensional Homotopic Compaction", *Algorithmica*, vol.6, pp.103-128, 1991.

[Male 96] F. M. Maley, "Testing Homotopic Routability Under Polygonal Wiring Rules", *Algorithmica*, Vol 15, pp.1-16, Jan. 1996.

[MaSa 95] Malgorzata Marek-Sadowska, and Majid Sarrafzadeh, "The Crossing Distribution Problem," *IEEE Trans. Computer-Aided Design*, vol. CAD-14, no. 4, pp.423-433, Apr. 1995.

[MeNa 90] K. Mehlhorn and S. Näher, "A Faster Compaction Algorithm with Automatic Jog Insertion", *IEEE Trans. on Computer-Aided Design*, Vol 9.,pp.451-466, Feb. 1990.

[Ohts 86] edited by T. Ohtsuki, "Layout Design and Verification", Advances in CAD for VLSI, vol. 4, North-Holland, 1986.

[Oust 84] J. K. Ouserhout, G. Handachi, R. N. Mayo, W. S. Scott and G. S. Taylor, "The Magic VLSI Layout System," in *Proc. 21st Design Automation Conf.*, June, 1984.

[Pin 82] Pinter, "River Routing: Methodology and Analysis", *3rd Caltech Conf. on VLSI*, pp.141-164, 1982.

[PrSh 85] F. Preparata and M. Shamos, *Computational Geometry, An introduction*, Spring-Verlag, 1985.

[Rich 84] D. Richards, "Complexity of Single-Layer Routing", *IEEE Trans. on Computers*, Vol. 33, pp. 286-288, Mar. 1984.

[SL 89] M. Sarrafzadeh and D. T. Lee, "A New Approach to Topological Via Minimization", *IEEE Trans. on Computer-Aided Design of Integrated Circuits*, vol.8, pp.890-900, Aug., 1989.

[SaLi 94] M. Sarrafzadeh, K.-F. Liao, and C. K. Wong," Single-Layer Global Routing", *IEEE Trans. on Computer-Aided Design*, Vol 13, pp. 38-47, Jan., 1994.

[SaPa 82] S. Sastry and A. Parker, "The Complexity of 2-Dimensional Compaction of *VLSI* Layouts", *Proc. IEEE International Conference on Circuits & Computers*, New York, 1982, pp. 402-406.

[SLW 83] M. Schlag, Y. Z. Liao and C. K. Wong, "An Algorithm for Optimal Two-Dimensional Compaction of VLSI Layouts", *Integration, VLSI journal*, pp. 179-209, 1983.

[Sott 84] W. S. Scott and J. K. Ouserhout, "Plowing: Interactive Stretching and Compaction in Magic,", *Proc. 21st Design Automation Conf.*, pp. 166-172, June, 1984.

[ShLo 89] H. Shin and C.-Y. Lo,"An Efficient Two Dimensional Compaction of VLSI Layouts", *Proc. of the 26th Design Automation Conf.*, pp.290-295, 1990

[WoMa 88] W. H. Wolf, R. G. Mathews, J. A. Newkirk, and R. W. Dutton, "Algorithms for Optimizing Two Dimensional Symbolic Layout Compaction", *IEEE Trans. on Computer-Aided Design*, Vol. 7, pp.451-466, Apr. 1988.