Minimizing Channel Density by Lateral Shifting of Components

D. S. Johnson[1] A. S. LaPaugh[2] R. Y. Pinter[3]

Abstract

We consider a channel layout model in which we are free to change the spacing, but not the ordering, of the components on the top and bottom of a routing channel. The goal is to minimize channel density. Let N be the total number of terminals on top and bottom components. The main previous approach to this problem used dynamic programming and restricted the potential locations for components and their terminals to grid points along a one-dimensional line, yielding running times polynomial in L, where L is the length of the grid. To use such an algorithm in an environment where high-precision placement is possible, however, one must have a very fine grid, with L growing as $\Omega(N^2)$ or worse. Consequently, the running time could grow very large as a function of N. We prove a key combinatorial lemma that implies that the run time for a grid-free dynamic programming algorithm can be polynomially bounded. Based on this lemma, a naive $O(N^5)$ algorithm is derived. We then show how this can be sped up to $O(N^3)$ by computing appropriately defined auxiliary functions. For certain instances of the problem in which all components are roughly the same size and have roughly $\sqrt{N}$ terminals, we can further reduce the running time to $O(N^{2.5})$.

1. Introduction

This paper concerns the interaction between placement and routing in the context of channel routing. In channel routing, the terminals to be connected lie on components placed along two parallel lines—the top and bottom of a common channel. See Figure 1. The sets of terminals to be connected are called *nets*. The connections are to be made using wires of a given, non-negligible width, and wires must be routed within the space between the two lines (the channel). Typically a routing that minimizes the distance needed between the two sides (the channel width) is desired. The quality of the routings that can be obtained will of course depend on the locations of the components along the top and bottom of the channel. To the extent that designers are free to shift these locations, they may be able to improve on the channel widths they can attain.

In this paper we consider the situation in which the identities and the orderings of the components along each side of the channel are fixed, but the exact positions (the lateral placement) of the components may vary. That is, components can slide back and forth along their side of the channel, but may not jump over one another or switch sides. Within each component, the terminals lie at fixed positions relative to the component boundaries. Considering this lateral flexibility is useful for refining a placement.

We would like to find lateral placements that minimize the channel width required for routing. Computing the minimum channel width given a lateral placement is not always feasible, however. (The most notable exception is for the river routing model, where a lateral placement that minimizes the channel width can be found in time $O(N\log N)$ [LePi83].) For instance, in the popular Manhattan wiring model, the problem of finding a minimum-width routing for a particular placement of terminals along the sides of a channel is NP-hard ([Szy85]). Even computing the channel width required by a particular routing algorithm usually requires execution of the routing algorithm, which is usually too time-consuming to do repeatedly during placement. Thus our goal shall instead be to minimize channel *density*. This is the maximum number of nets that can be cut by a vertical line through the channel, where we say a net is *cut* by a vertical line if the line passes through a terminal

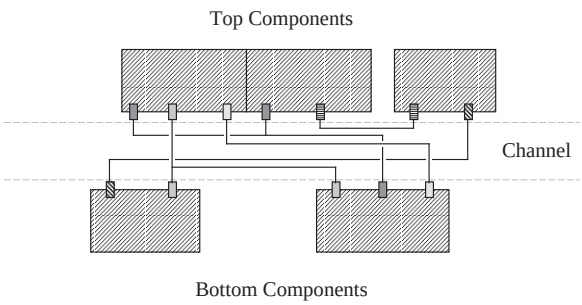

FIGURE 1. A routing channel, with terminals color-coded by nets.

[1] Room 2D-150, AT&T Bell Laboratories, Murray Hill, NJ 07974

[2] Department of Computer Science, Princeton University, Princeton, NJ 08544. Research sponsored in part by AT&T Bell Laboratories, by the National Science Foundation under grants MCS-8202594 and MIP-86119335 and by DARPA/ONR under contract N00014-88-K-0459.

[3] IBM Scientific Center, Haifa, Israel. Research performed in part while employed by AT&T Bell Laboratories.

belonging to the net, or if the net contains one or more terminals on each side of the line. Density can be computed quickly, and it provides a fairly accurate measure for the minimum width that can be achieved under a variety of routing models, including Manhattan and knock-knee [LaPi90]. Although density is not the only lower bound for channel width ([BrRi81], [BBL84]), in most practical situations it is a good measure for estimating the width during placement evaluation and rough routing (see, for example, [Pe76]). The situations in which the other lower bounds exceed density are extremely rare. In fact, many existing channel routers achieve widths that are usually within one of the density ([Deu76], [FiRi82], [BuPe83]); the cases in which other measures become dominant are easy to recognize and can be dealt with separately.

Variants and special cases of this problem has been studied before. LaPaugh and Pinter [LaPi83] and independently Atallah and Hambrusch [AtHa86] have shown that if there are just two components, one on the top and one on the bottom of the channel, density can be minimized in time $O(N^2 \log N)$, where N is the total number of terminals. (Essentially, there are just $O(N^2)$ relative placements of the two components to consider.) Gopal et al. [GCW83] consider the case where no component contains more than one terminal, providing an $O(N^2)$ algorithm.

The most general previous results are based on the work of Cai and Wong [CaWo91], whose results recently have been improved by Greenberg and Shih [GrSh93]. Cai and Wong extend the model of [GCW83] to allow optional constraints on the distances between successive terminals on the same side of the channel (i.e., one may require that the distance between terminal x and its successor be $\leq$, $=$, or $\geq$ some constant). Note that it is easy to model our problem using extra constraints of this sort. Cai and Wong then add the restriction that terminals must be located at integral positions along the channel, and propose a grid-based dynamic programming algorithm, whose running time for our problem is $O(L^3)$, where L is the length of the channel. Greenberg and Shih improve this running time to $O(L^2 \log N)$. The restriction to integral positions is not a major constraint, and indeed is satisfied by many technologies. However, the scale of the grid may have to be very fine if it is to accurately reflect real terminal locations, even on a single component. Consequently, L may have to be very much bigger than N. That is, at the resolution needed to represent the positions of terminals, the number of terminals may be very sparse with respect to the lengths of components. For instance, a value for L that is at least $\Omega(N^2)$ is implicit in the analyses of the two-component case mentioned above. Thus the running time of Greenberg and Shih's algorithm, when restated in terms of the number of terminals, could well be $\Omega(N^4 \log N)$, if indeed it is bounded in terms of N at all.

In this paper we show that our general lateral placement problem can be solved in polynomial time without discretization. Without a grid to provide reference points, we first must find a way to represent channel density as a function of component positions. We choose to use the terminals and endpoints of the set of bottom components as reference points. A pair of these points delimits a portion of the channel, and we define a density function that varies as different positions on top components are aligned with the reference points on the bottom. The points in the domain correspond to pairs (p,q), where each element is either a special symbol representing the potential gap between a pair of consecutive components, or else a point on a component (arbitrary interior points allowed, not just endpoints and terminals). We are confronted (at least initially) with the problem of representing functions whose domains are two-dimensional and uncountable, and whose values at any given point involve a separate density computation, at least in principle. Fortunately, we can show by a combinatorial argument that such a domain can be partitioned into a polynomial number of points, line segments, rectangles, and triangles, on each of which the density function is constant. Based on this insight, a polynomial-time dynamic programming algorithm for finding a minimum-density placement becomes possible.

In Section 2 we shall introduce the notation and basic framework for our dynamic programming approach, and state the key lemma that insures that the functions used in this framework have polynomial-length descriptions. Section 3 then provides key details of our algorithms. We begin with a fairly simple $O(N^5)$ algorithm based on the lemma. We then outline the data structures and auxiliary functions we have devised to reduce the overall running time to $O(N^3)$ and briefly indicate the additional ideas that allow us to reduce the time to $O(N^{2.5})$ in the special case when both the number of components and the maximum number of terminals per component is $O(\sqrt{N})$. We conclude in Section 4 with a discussion of variants on our problem, subsequent work that has been based on our results, and directions for future research.

2. Preliminaries and a Key Lemma

We model components as straight line segments, and terminals by points on those line segments. Given the definition of density and the technological constraint that any two terminals on the same side of the channel must be placed at least some minimum distance apart, we may assume without loss of generality that (a) each net contains at most 4 terminals (two each on top and bottom), (b) each component contains at least one terminal, (c) no terminal is an endpoint of its component,, and (d) for any pair of consecutive components on the same side of the channel, any gap $g \geq 0$ between them is allowable in a placement ($g = 0$ implies the two components *abut*, i.e., their respective left and right endpoints coincide). If an instance does not satisfy these constraints, in can be transformed to one that does in a linear-time preprocessing step that may delete terminals,

merge components, or add additional terminal-free segments to the ends of a component.

Now for some notation: Let T_b (T_t) be the set of terminals of bottom (top) components, and let $N_b = |T_b|$ and $N_t = |T_t|$. Let c_b (c_t) be the number of bottom (top) components, and label those components as $K_1^b, K_2^b, \ldots, K_{c_b}^b$ ($K_1^t, K_2^t, \ldots, K_{c_t}^t$), respectively. Let $bl(i)$ and $br(i)$ be the left and right endpoints of bottom component K_i^b, and define $tl(i)$ and $tr(i)$ analogously for top component K_i^t. Let E_b (E_t) be the set of endpoints of bottom (top) components, augmented by *points-at-infinity* $br(0)$ and $bl(c_b+1)$ ($tr(0)$ and $tl(c_t+1)$). Let A_t be the set of all the points on the line segments representing top components (not just the terminals and endpoints, but the whole uncountable set), augmented by the two top points-at-infinity. Finally, let G_t be a collection of c_t+1 top *gap points*, denoted $tr(i) + \Delta$, $0 \le i \le c_t$.

Let us call $E_b \cup T_b$ the set of *bottom alignment points* and $A_t \cup G_t$ the set of *top alignment points*, and let "$<_b$" and "$<_t$" be the natural orders defined on each set of alignment points by $x <_b y$ ($x <_t y$) if x and y are both on the bottom (top) side of the channel and x is to the left of y in the given ordering of components for that side. If $l <_b r$ are bottom alignment points, we shall say that $[l,r]$ is a *bottom alignment interval*.

The following function is the basis for all our dynamic programming algorithms. Let $[l,r]$ be a bottom alignment interval and let $p \le_t q$ be members of $A_t \cup G_t$ ($p = q$ is allowed). Then $D_{l,r}^{p,q}$ is the minimum, over all placements P in which p is aligned with l and q is aligned with r, of the maximum density between l and r (inclusive) under P, with $D_{l,r} = \infty$ if no placement is possible with the given alignment. (This happens, for instance, when p and q must be further apart than l and r can be.) If q is the gap point $tr(i) + \Delta$, we restrict attention to placements where there is a non-zero gap between components K_i^t and K_{i+1}^t and bottom point r lies in this gap; if p is a gap point, an analogous restriction is imposed with respect to l. In our algorithms, we shall be treating l and r as (temporarily) fixed, so that $D_{l,r}^{p,q}$ becomes simply a function of p and q (which we shall denote by $D_{l,r}^{*,*}$ to distinguish the function from its value for the pair (p,q)).

The main problem confronting us is that of obtaining compact representations for the functions $D_{l,r}^{*,*}$. We do this by constructing, for each (l,r), a partition of two-dimensional (p,q)-space (the domain of $D_{l,r}^{*,*}$) into regions on which $D_{l,r}^{*,*}$ is constant. Our major technical result is that regions on which $D_{l,r}^{*,*}$ is constant are determined by a polynomial number of vertical and horizontal lines and two diagonal lines in (p,q)-space. The regions are the lines segments, points, rectangles, and triangles into which these lines partition (p,q)-space. See Figure 2, which illustrates the partition when l and r are on the same component. The shaded regions ar those for which $D_{l,r}^{*,*} < \infty$. (When l and r are on different components, the picture is similar, but now all of (p,q)-space above the upper diagonal is shaded.) This partition will enable us to construct a compact representation for each function $D_{l,r}^{*,*}$ and, based on these representations, to develop efficient dynamic programming algorithms for computing the optimal density.

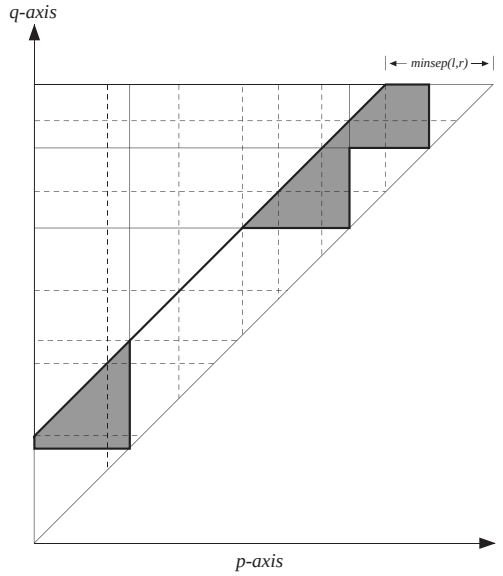

FIGURE 2. (p,q)-space when l and r are on the same component.

We first note that if one only wants to separate the infinite-valued regions of (p,q)-space from the finite-valued regions, finding the required bounding lines is not difficult. Beyond the two diagonal lines mentioned above, we will need only those horizontal and vertical lines that pass through the endpoints of the top components (represented by solid lines in Figure 2). As to the diagonal lines, one is the $q = p$ line, since we are restricting attention to those pairs p,q with $p \ge_t q$. The second diagonal is parallel to the first, and to justify it we will need two more definitions and a lemma.

Definition. *Let s and t be points on the same component (top or bottom), with s to the left of t. Then* $\mathrm{sep}(s,t)$ *is the (fixed) distance between the two points, measured along the component.*

Definition. *Let s and t be component points on the same side of the channel, with s to the left of t. Then* $\mathrm{minsep}(s,t)$ *is the minimum possible distance between the two points under any placement. More precisely, suppose s and t are on bottom components (the definition is analogous if they are on top components). If s and t are on the same component, minsep*$(s,t) = sep(s,t)$*. Otherwise, suppose s is on component* K_i^b *and t is on* K_j^b, $i < j$. *Then* $\mathrm{minsep}(s,t) =$

$$sep(s,br(i)) + \sum_{h=i+1}^{j-1} sep(bl(h),br(h)) + sep(bl(j),t).$$

Note that this is the distance between s and t in the *minimum separation configuration* in which no gaps are left between any of the components on the given side of the channel.

Lemma 2.1. *Suppose $p \leq_t q$ are points on top components (i.e., not gap points). Then $D^{p,q}_{l,r} < \infty$ if and only if*
(a) *l and r are on different components, as are p and q, or*
(b) *l and r are on the same component, p and q are on different components, and $minsep(p,q) \leq sep(l,r)$, or*
(c) *l and r are on different components, p and q are on the same component, and $minsep(l,r) \leq sep(p,q)$.*
(d) *l and r are on the same component, p and q are on the same component, and $sep(l,r) = sep(p,q)$.*

The last three constraints imply that the finiteness of of $D^{*,*}_{l,r}$ at a given point in (p,q)-space may depend on whether that point is above, below, or on the line $q = p + minsep(l,r)$, and this is the second diagonal line mentioned above. In the full paper, the above Lemma is extended to cover pairs (p,q) in which one or both of p and q are gap points, since these pairs are also distinct members of ''(p,q)-space.'' Note that in Figure 2, the single x-coordinate $tr(1)$ also represents $tr(1)+\Delta$ and $tl(2)$, all forced to coincide because we are considering minimum separation configurations.

To show that the finite-valued regions of (p,q)-space can be partitioned into constant-valued regions by a polynomial number of additional vertical and horizontal lines (dotted in Figure 2) is more difficult. Given only one top component and one bottom component, the number of changes of the value of channel density can equal the number of *topologically distinct* placements, of which there can be $\Theta(N^2)$, as observed in [LaPi83,AtHa86]. (Two placements are topologically distinct if they give rise to different orderings, alignments included, of the terminals along the channel. Note that density depends on this ordering rather than the exact positions of terminals.) In the multi-component case, the number of topologically distinct placements can be exponential: a pair of top and bottom components that overlap can interact independently of other pairs in a placement. To show that the number of constant-valued regions in the domain of $D^{*,*}_{l,r}$ is polynomial, we shall define *potential points of change*, show that the number of such points is polynomial, and show that the regions of constant value in (p,q)-space can be bounded by our two diagonals together with the vertical and horizontal lines through the potential points of change.

We now present some of the detail of our technique. Let a placement be called a *minimum separation placement* if both the top and the bottom of the channel are in minimum separation configurations. Note that for any alignment of r with a top point, there is exactly one minimum separation placement.

Definition. *Let l and r be members of $T_b \cup E_b$. The set of* potential right points of change for $[l,r]$, *denoted by $R_{l,r}$, consists of all points q on top components for which there exist points s and x having the following properties:*
(a) *$s \in E_b \cup T_b$ lie between l and r (inclusive),*
(b) *$x \in E_t \cup T_t$ and lies to the left of q or equals q,*
(c) *Either s is on the same component as r or x is on the same component as q, and*
(d) *$minsep(s,r) = minsep(x,q)$, i.e., points s and x are aligned in the minimum separation placement that aligns q with r.*

Note that $E_t \cup T_t \subseteq R_{l,r}$, by definition. The set of *potential left points of change for* $[l,r]$ is defined analogously.

Lemma 2.2. *Suppose $[l,r]$ is a bottom alignment interval and let $m_{l,r} = |[l,r] \cap (E_b \cup T_b)|$ Then $|L_{l,r}| = O(N_t \cdot m_{l,r})$ and $|R_{l,r}| = O(N_t \cdot m_{l,r})$. This upper bound is $O(N^2)$ in general and $O(N)$ when l and r are consecutive bottom alignment points.*

Proof. Each potential left or right point of change derives from a possible alignment of a member of $E_t \cup T_t$ with a member of $E_b \cup T_b$ lying in the closed bottom interval bounded by l and r. At most one left (right) point of change can derive from each such alignment. The set $E_t \cup T_t$ is of size $O(N_t) = O(N)$ by assumption (R4) of Section 2, and the set $E_b \cup T_b$ is of size $O(N_b) = O(N)$. □

Definition. *The (l,r)-*partition *of p,q-space is that induced by all the horizontal and vertical lines whose fixed p-coordinates (q-coordinates) are gap points or members of $L_{l,r}$ ($R_{l,r}$). This partition divides the points of p,q-space into three types of sets: single-element sets (*intersection points*) representing the intersections of the above lines, open line segments (*segments*) representing the parts of the lines between consecutive intersection points, and open* regions *(triangles, rectangles, and an occasional pentagon) bounded by the above lines (and possibly the $p = q$ line) but containing no points from any such lines.*

We claimed above that $D^{*,*}_{l,r}$ would be essentially constant on all the segments and regions of the partition. We said ''essentially'' because for one particular type of region we have to settle for two values, one finite and one infinite. To explain this, we first introduce the concept of a *critical pair*.

Definition. *A pair p,q of top alignment points with $p <_t q$ is a* critical *pair for a bottom alignment interval $[l,r]$ if and only if $minsep(p,q) = minsep(l,r)$ and either l and r are on the same bottom component or p and q are on the same top component.*

Note that critical pairs lie on the $q = p + minsep(l,r)$ line, although only certain segments of that line will consist of critical pairs when l and r lie on different components. In light of Lemma 2.1, the question of whether a segment or region contains a critical pair is key to the question of whether $D^{*,*}_{l,r}$ can take on both finite and infinite values in that segment or region. The next two lemmas describe how the presence of critical pairs in segments and regions is constrained. The first implies that no segment can contain a critical pair, since critical points on lines of potential change can only occur at intersection points of the partition.

Lemma 2.3. *If (p,q) is a critical pair for a bottom alignment interval $[l,r]$, then $p \in L_{l,r}$ if and only if $q \in R_{l,r}$.*

Proof. (Not difficult, omitted.)

Lemma 2.4. *Let U be an open region of the (l,r)-*

partition consisting of pairs (p,q) satisfying $p < q$, $p_0 < p < p_1$, and $q_0 < q < q_1$, where p_1 and q_1 are the immediate successors of p_0 and q_0 in $L_{l,r}$ and $R_{l,r}$ respectively. Then if U contains a critical pair, that pair lies on the diagonal line segment between (p_0,q_0) and (p_1,q_1), and all pairs on that line segment are critical pairs.

Proof. (Not difficult, omitted.)

We shall call the line segment described in Lemma 2.4 a *critical diagonal*, and the open region region containing it a *critical region*. Reinterpreting Lemma 2.1 in terms of Lemmas 2.3 and 2.4 and the above definitions, we have

Theorem 2.1. *For any line segment and for any region U of the (l,r)-partition that does not contain a critical pair, $D^{*,*}_{l,r}$ is either finite for all pairs in U or infinite for all pairs in U. If a region U contains a critical pair, it has a critical diagonal, and the division of its pairs into those with finite and infinite values for $D^{*,*}_{l,r}$ is as follows:*

(a) *If U is a same-component region (for all pairs (p,q) in U, p and q are on the same component) and l and r are on the same component, then $D^{*,*}_{l,r} < \infty$ only on the critical diagonal.*

(b) *If U is a same-component region and l and r are on different components, then $D^{*,*}_{l,r} < \infty$ only for points of U on or above the critical diagonal.*

(c) *If U is a different-component region and l and r are on the same component, then $D^{*,*}_{l,r} < \infty$ only for points of U on or below the critical diagonal.*

Our goal is to prove that $D^{*,*}_{l,r}$ is constant on each line segment of the (l,r)-partition and on each region U that does not contain a critical pair, and that for a region U that does contain a critical pair, $D^{*,*}_{l,r}$ takes on only one non-infinite value. The proof of this depends on the following key technical result (as well as the analogous result for left points of change, proved similarly).

Lemma 2.5. *Suppose $[l,r]$ is a bottom alignment interval and q_1 and q_2 are two consecutive points in $R_{l,r}$ that are on the same component. Then for any top positions x_1 and x_2 such that $q_1 < x_1 < x_2 < q_2$ and for all top alignment points $p \le x_1$ such that both $D^{p,x_1}_{l,r} < \infty$ and $D^{p,x_2}_{l,r} < \infty$, we have $D^{p,x_1}_{l,r} = D^{p,x_2}_{l,r}$.*

Proof. The proof is by contradiction, so assume $D^{p,x_1}_{l,r} \neq D^{p,x_2}_{l,r}$. There are two cases, depending on whether $D^{p,x_1}_{l,r} > D^{p,x_2}_{l,r}$ or whether $D^{p,x_1}_{l,r} < D^{p,x_2}_{l,r}$.

Our general approach will be to transform an optimal configuration in which x_2 is aligned with r into one in which x_1 is aligned with r, or vice versa. We do this by sliding a top or bottom component (among those components overlapping the $[l,r]$ interval in the configuration) either leftward or rightward. When sliding such a component left, it may collide with a second component before we are finished moving it. In this case, we start sliding both components simultaneously, keeping them in contact with each other (abutting); we can think of this as the first component pushing the second. Again, we may collide with a third component before we reach the conditions we want, and we continue moving all three. In this way we may move an ever-increasing sequence of abutting adjacent components leftward. We will refer to this as ''pushing a component and subsequent abutting components.''

Let K^t_i be the component that contains q_1 and q_2, and hence also x_1 and x_2.

Case 1: $D^{p,x_1}_{l,r} > D^{p,x_2}_{l,r}$.

We wish to transform an optimal configuration for x_2 into one for x_1. We do this by pushing the bottom component containing r to the left, while keeping the positions of all top components fixed. We begin with any configuration achieving $D^{p,x_2}_{l,r}$ and push the bottom component containing r and subsequent abutting components until the first time at which one of the following list of alternatives occurs, with ties broken according to the order of the list:

(1) The density of the configuration in the interval $[l,r]$ changes, or

(2) Point r is aligned with x_1, or

(3) If p is on K^t_i (i.e., $p \ge tl(i)$), all bottom components from the one containing r to the one containing l abut (see Figure 3a, or

(4) If $p < tl(i)$, the sequence of abutting components going left from the one containing r extends at least to the position of $tl(i)$ in the configuration (see Figure 3b).

First note that alternative (2) will eventually occur if none of the others do, so we can be assured the pushing process will terminate. What we must show is that each option either implies a contradiction directly or else implies the existence of a configuration in which l is aligned with p, x_1 is aligned with r, and the overall density is equal to $D^{p,x_2}_{l,r}$, contradicting the basic assumption of this Case that $D^{p,x_1}_{l,r} > D^{p,x_2}_{l,r}$.

Suppose (1) is the first to occur. Since the density of a configuration between l and r is completely determined by the combined order of appearance of top and bottom terminals between the positions of l and r, with alignments taken into account, the change must have occurred when some top and bottom terminals changed their relative positions, say x on top and s on the bottom. For the terminals to have

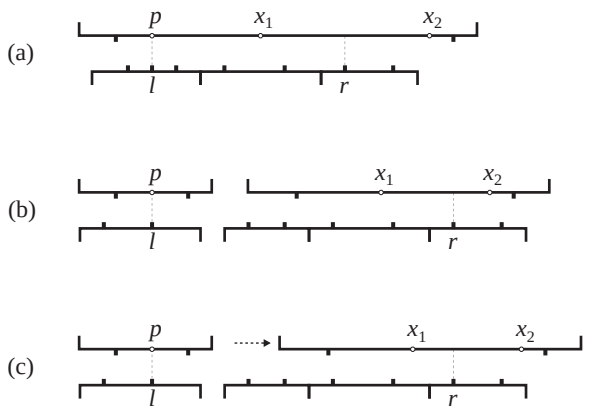

FIGURE 3. Illustrations for proof of Case 1 of Lemma 2.5.

changed their relative positions, they must have gone into or out of alignment. The fact that neither alternative (3) nor (4) has occurred previously insures that x must be located on K_i^t and s must be located on the sequence of contiguous components being pushed on the bottom. See Figure 3a. Let q be the top point that is aligned with r when x is aligned with s. Then we have $sep(x,q)=minsep(s,r)$ and so $q \in R_{l,r}$ by definition. Furthermore, since (2) has not previously occurred, we must also have $x_1 \le q \le x_2$. By our initial hypothesis about x_1 and x_2, this means there is an element of $R_{l,r}$ strictly between q_1 and q_2, contradicting our choice of q_1 and q_2.

Suppose we stop because of (2), and (1) does not occur simultaneously. Then we have found a configuration in which x_1 is aligned with r, p is aligned with l, and the density is $D_{l,r}^{p,x_2}$, the desired contradiction mentioned above.

Suppose we stop because of (3) and neither (1) nor (2) occurs simultaneously. Then r must be aligned with a top position x to the right of x_1 for which $sep(p,x) = minsep(l,r)$. Then $sep(p,x_1) < sep(p,x) = minsep(l,r)$. But by Lemma 4.1(1c) this implies $D_{l,r}^{p,x_1} = \infty$, contrary to hypothesis.

Finally, suppose we stop because of (4) and neither (1) nor (2) occurs simultaneously. In this case we freeze the positions of the bottom components and begin to push component K_i^t to the right, proceeding until either the density of the configuration in the interval $[l,r]$ changes or x_1 is aligned with r. Note that during this pushing process, x_1 will never encounter a gap between components on the bottom, since by (4) the bottom of the channel is now a solid mass of abutting components from r to the left end of K_i^t. See Figure 3c. If the density changes first (or changes simultaneously with the alignment of x_1 with r), we have the same conclusions as for alternative (1) above. If we succeed in aligning x_1 with r without changing the density, we have the same conclusions as for condition (2) above. In both cases we have the needed contradiction.

Case 2: $D_{l,r}^{p,x_1} < D_{l,r}^{p,x_2}$.

The proof is similar to that for Case 1, although this time we must transform an optimal configuration for x_1 into one for x_2. We do this by pushing a top component instead of a bottom one, which makes part of the argument a bit more complex. Because of space limitations, we leave the proof for the full paper. □

We are now ready to prove our main theorem about the functions $D_{l,r}^{*,*}$.

Theorem 2.2. *$D_{l,r}^{*,*}$ is constant on each line segment of the (l,r)-partition and on each region U that does not contain a critical pair. For a region U that does contain a critical pair, $D_{l,r}^{*,*}$ takes on only one non-infinite value.*

Proof. In light of Theorem 2.1, there are five basic cases: (a) segments of the (l,r)-partition, (b) regions U of the partition not containing a critical diagonal, (c) regions U containing a critical diagonal and having $D_{l,r}^{*,*} < \infty$ for some point above that diagonal, (d) regions U containing a critical diagonal and having $D_{l,r}^{*,*} < \infty$ for some point below that diagonal, and (e) regions U containing a critical diagonal and having $D_{l,r}^{*,*} < \infty$ only for points on that diagonal.

Case (a) involves a straightforward application of Lemma 2.5. Suppose the segment is vertical. By Theorem 2.1, if any point in the segment has $D_{l,r}^{*,*} = \infty$ they all do, so in this case all have the same value. Otherwise, let (p,x_1) and (p,x_2) be two distinct points on the segment, $x_1 < x_2$. By definition of the (l,r)-partition, there also must exist consecutive points q_1 and q_2 in $R_{l,r}$, such that $q_1 < x_1 < x_2 < q_2$. (The non-included ''endpoints'' of the open segment will be (p,q_1) and (p,q_2).) Thus the desired conclusion that $D_{l,r}^{p,x_1} = D_{l,r}^{p,x_2}$ follows by an application of Lemma 2.5(a). The case of horizontal segments is handled analogously using Lemma 2.5(b).

Cases (b), (c), and (d) are all handled similarly, each using one application each of Lemma 2.5(a) and Lemma 2.5(b). For example, suppose we are in case (c), in which case we know by Lemma 2.1 that for $(x,y) \in U$, $D_{r,succ(r)}^{x,y} < \infty$ if and only if (x,y) is on or above the critical diagonal of U. Let (x_1,y_1) and (x_2,y_2) be two pairs in U with $D_{l,r}^{*,*} < \infty$. Without loss of generality we may assume that $x_1 \le x_2$. If $y_1 \le y_2$, let $x_3 = x_1$ and $y_3 = y_2$. Otherwise, let $x_3 = x_2$ and $y_3 = y_1$. In both cases, (x_3,y_3) will be in U and we will have $D_{l,r}^{x_3,y_3} < \infty$. But then with one application each of Lemma 2.5(a) and Lemma 2.5(b) we obtain $D_{l,r}^{x_1,y_1} = D_{l,r}^{x_3,y_3} = D_{l,r}^{x_2,y_2}$. Thus any two points in U with $D_{l,r}^{*,*} < \infty$ have the same value for $D_{l,r}^{*,*}$, as required.

Finally, for case (e), the only points in U with $D_{l,r}^{*,*} < \infty$ are those on the critical diagonal. Let (x_1,y_1) and $(x_{2,}y_2)$ be two distinct pairs on that diagonal, with $x_1 < x_2$. By Theorem 2.1, $D_{l,r}^{*,*} < \infty$ for both pairs. We must show that $D_{l,r}^{x_1,y_1} = D_{l,r}^{x_2,y_2}$. Since these are critical pairs, we know that $minsep(x_1,y_1) = minsep(x_2,y_2) = minsep(l,r)$. By Theorem 2.1 we also know that l and r are on the same bottom component K_j^b and x_1, x_2, y_1 and y_2 are on the same top component K_i^t. Thus for each $i \in \{1,2\}$ there is a unique placement that aligns l with x_i and r with y_i. In each case, $D_{l,r}^{x_i,y_i}$ will be precisely the density of this configuration in the interval $[l,r]$.

So suppose we start with the placement of minimum separation configurations that aligns l and x_1 (and determines $D_{l,r}^{x_1,y_1}$) and then slide the top component K_i^t left until l aligns with x_2. As in the proof of the Lemma 2.5, the density between l and r cannot change unless a terminal alignment is encountered. By a now-familiar argument, however, this would imply the existence of a member of $R_{l,r}$ between y_1 and y_2, and so (x_1,y_1) and (x_2,y_2) could not be in the same region of the partition, contrary to our hypothesis that they are on the critical diagonal of a single region. Thus when we arrive at the alignment of l and x_2 (which determines $D_{l,r}^{x_2,y_2}$) we will still have our original density. We thus conclude as required that $D_{l,r}^{x_1,y_1} = D_{l,r}^{x_2,y_2}$.

Since Cases (a) through (e) exhaust the possibilities, this concludes the proof of Theorem 2.2. □

As a consequence of Theorems 2.1 and 2.2, a complete description of $D^{*,*}_{l,r}$ need contain only one value for each point, segment, and region of the (l,r)-partition. We conclude this section by introducing a shorthand notation that will allow us to concisely represent the sets of the partition for which distinct density values need to be computed. The basis for this shorthand consists of the sets $L^{\theta}_{l,r}$ and $R^{\theta}_{l,r}$, with the latter set consisting of the set $R_{l,r}$ augmented by the pseudo-points $tr(0)$ and $tl(c_t+1)$, the gap points $tr(i)+\Delta$, and by *interval points* $q+\theta$ for each $q \in R_{l,r}$ that is not the right endpoint of a component. ($L^{\theta}_{l,r}$ is defined analogously.)

The set $\Pi_{l,r} = L^{\theta}_{l,r} \times R^{\theta}_{l,r}$ then can be taken to represent all the regions, segments, and intersection points of the (l,r)-partition (plus some ''non-regions'' which can be identified as such and will be ignored in what follows). In summary, Theorem 2.2 tells us that $D^{*,*}_{l,r}$ can be completely described by storing one value for each pair $(x,y) \in \Pi_{l,r}$, along with a notation as to which pairs correspond to critical regions. In the special case where we are only interested in the values of $D^{p,q}_{l,r}$ for some fixed value of p, a case that plays a major role in our algorithm, the description can be even simpler, as we need only store the values of $D^{p,q}_{l,r}$ for each $q \in R^{\theta}_{l,r}$.

3. Algorithm Sketches

We shall sketch three different algorithms, each of the latter two yielding an improvement in efficiency over its predecessors. All three follow a general scheme in which we compute the functions $D^{*}_{v} \equiv D^{tr(0),*}_{br(0),v}$ for each $v \in E_b \cup T_b$ in turn, starting with $D^{*}_{bl(1)}$ and proceeding one by one until we reach $D^{*}_{bl(c_b+1)}$, whose minimum value equals the minimum possible density. If $v \in E_b \cup T_b$ let $succ(v)$ denote the member of $E_b \cup T_b$ immediately to the right of v. In the first two algorithms, we compute $D^{*}_{succ(v)}$ from the previously computed function D^{*}_{v} and the independently computed function $D^{*,*}_{v,succ(v)}$.

ALGORITHM 1. Our first algorithm is a straightforward application of the lessons learned in Section 2. We begin by constructing the sets $R^{\theta}_{v} = R^{\theta}_{br(0),v}$, which we can do for all $r \in E_b \cup T_b$ in overall time $O(N^3)$ time by straightforward techniques. To compute the functions $D^{*,*}_{v,succ(v)}$, we first note that there are at most $O(N)$ right and left potential points of change by Lemma 2.2. Moreover, the value of the function for all such pairs can be computed in overall time $O(N^2)$ if organized properly. This is because there are no bottom terminals strictly between v and $succ(v)$. Thus the maximum density obtainable in the $[v,succ(v)]$ interval when x is aligned with v and y is aligned with $succ(v)$ is independent of the placement within that interval of any top components that might lie between the ones containing x and y. Consequently a complete representation of $D^{*,*}_{v,succ(v)}$ can be computed in time $O(N^2)$, for a total cost of $O(N^3)$ over all pairs $(v,succ(v))$.

The bottleneck is the computation of $D^{*}_{succ(v)}$ from D^{*}_{v} and $D^{*,*}_{v,succ(v)}$. Let $R^{\theta}_{v} = R^{\theta}_{br(0),v}$. It is not difficult to show that the sets R^{θ}_{v} for all v in $E_b \cup T_b$ can be identified in total time $O(N^3)$. Unfortunately, for any y in $R^{\theta}_{succ(v)}$, the value of $D^{y}_{succ(v)}$ would seem to derive from a minimization over $O(N^2)$ terms, one for each of the $O(N^2)$ elements x of R^{θ}_{v} and one for each of the $O(N)$ left points of change x for $D^{*,*}_{v,succ(v)}$.

$$D^{y}_{succ(v)} = \min_{x} \left\{ \max \left[D^{x}_{v}, D^{x,y}_{v,succ(v)} \right] \right\}$$

ALGORITHM 1 uses the above relation in the straightforward way to compute a representation of $D^{*}_{succ(v)}$ from that for D^{*}_{v} in time $O(N^4)$. Since there are $O(N)$ possible values for v to be considered, this gives us a $O(N^5)$ overall running time bound.

ALGORITHM 2. To speed our basic algorithm up to $O(N^3)$ we compute several auxiliary functions based on D^{*}_{v} before going on to compute $D^{*}_{succ(v)}$. If x is a point on a top component but not in $E_t \cup T_t$, let $next(x)$ denote the member of $E_t \cup T_t$ that is closest to x among those that are strictly to its right, and let $prev(x)$ denote the member that is closest among those strictly to its left.

Definition. *Suppose $x < y$ are points on the same top component and v is in $E_b \cup T_b$. Then*

$$DMIN^{x,y}_{v} = \min\{D^{p}_{v} : x < p < y\}$$

$$DRMIN^{x}_{v} = DMIN^{x,next(x)}_{v}$$

$$DLMIN^{x}_{v} = DMIN^{prev(x),x}_{v}$$

The values of the latter two auxiliary functions, together with points $pRMIN^{x}_{v}$ and $pLMIN^{x}_{v}$ at which the corresponding minima are attained, can all be computed in time $O(N^2)$ given an appropriate data structure for D^{*}_{v}. The actual representationskk we use (for $DRMIN^{*}_{v}$ and $DLMIN^{*}_{v}$ as well as D^{*}_{r}) is a doubly-linked list of the members of R^{θ}_{r} in order, with the entry for an element q and giving the value of the given function for that element. Entries for members of $T_t \cup E_t$ also contain pointers to the entries for the next and previous members of $T_t \cup E_t$ under the $<_t$ order, and entries for gap points similarly contain pointers to the next and previous gap points.

The data structure of $D^{*,*}_{r,succ(r)}$ will be a doubly-linked list of lists, one for each element of $R^{\theta}_{r,succ(r)}$ in order. The sublist for $q \in R^{\theta}_{r,succ(r)}$ is a doubly-linked list of the values of the points $p \in L^{\theta}_{r,succ(r)}$ that yield finite values for $D^{p,q}_{r,succ(r)}$, in order, starting with the leftmost such p. (If the entry is for a critical rectangle, then the single finite value attained in that region is stored; our algorithm will only be accessing this value for pairs of top points that are on the region's critical diagonal and hence have $D^{*,*}_{r,succ(r)}$ equal to this value.) Within these lists each member of $T_t \cup E_t$ again points to its predecessor and successor in that set, as does each gap point. Assuming that we are given the sets

$R^{\theta}_{l,succ(r)}$ Note that such a representation can easily be constructed while computing the function $D^{*,*}_{r,succ(r)}$, all in $O(N_t^2) = O(N^2)$ time.

In order to compute $D^{*}_{succ(r)}$, we must by Proposition 2.2 determine its values for each element of $R^{\theta}_{succ(r)}$, so the first step is to identify this set and set up the doubly-linked list for it into which we eventually store the values of $D^{*}_{succ(r)}$. The straightforward method for generating all the sets R_r in one joint process that we use in ALGORITHM 1 does not generate the individual sets in sorted order. (Such ordering information was not needed for the overall $O(N^5)$ time bound derived there.) Since the sets can have $\Omega(N^2)$ items, we cannot afford to sort them after the fact (in time $\Omega(N^2 \log N)$) and still meet the claimed $O(N^2)$ bound for computing D^{*}_{r}. Fortunately, we can generate $R_{succ(r)}$ in sorted order in time $O(N^2)$ directly from the sorted representation for R_r, as we shall now show. (The initial set $R_{bl(1)}$ equals $T_t \cup E_t$, and so the data structure for it can be constructed in $O(N \log N)$, or indeed linear time, assuming as we have that the latter set was pre-sorted.)

As defined in Section 4, $R_{succ(r)}$ consists of $T_t \cup E_t$ together with any point q which, if aligned with $succ(r)$ in a minimum separation configuration for top and bottom, causes an alignment of a member u of $T_t \cup E_t$ with a member v of $T_b \cup E_b$, where either u is on the same component as q or v is on the same component as $succ(r)$. We can assume that we already have the points of $T_t \cup E_t$ in sorted order, so all we need do is construct an ordered list of the $q \in R_{succ(r)}$ that arise from the latter types of alignments and are not in $T_t \cup E_t$, and then merge the two lists. But since r and $succ(r)$ are successive members of $T_b \cup E_b$, the inducing alignment for such a q must take place at or to the left of r. This means that the point q' that is aligned with r in this configuration will be in R_r. Hence, we need only consider for this latter list points q that are not in $T_t \cup E_t$ and that have $minsep(q,q') = sep(r,succ(r))$ for some $q' \in R_r$. We can thus consider the elements q' of R_r, which we already have in sorted order, each of which gives rise to at most one candidate for $R_{succ(r)}$, also in sorted order. To verify whether the candidate is in fact a true member of $R_{succ(r)}$, we will also need to have stored the location of the rightmost u and v whose alignment caused q' to join R_r, but this can easily be arranged. Thus $R_{succ(r)}$ can be constructed in sorted order, and the initial doubly-linked data structure for $D^{*}_{succ(r)}$ can be built, all in time proportional to $|R_r| + n_t = O(N^2)$. While doing this we also add an auxiliary set of pointers to the $Dsucc(r)^{*,*}$ data structure that link each member of $R^{\theta}_{r,succ(r)}$ to the member of R^{θ}_{r} that represents it. (Note that by the definitions of the two sets, $R_{r,succ(r)}$ must be a subset of $R_{succ(r)}$, so that the latter yields a refinement of the partition induced by the former.) These auxiliary pointers can be constructed in time $O(N^2)$ by a parallel scan through the two sets. Having performed these constructions, we now proceed to the computation of $D^{*}_{succ(r)}$.

The work involved in computing $D^{*}_{succ(r)}$ from D^{*}_{r} breaks into two cases, depending on whether r and $succ(r)$ are on the same bottom component or not. We do not have space here to cover either case in complete detail, but will sketch the argument for the case when r and $succ((r)$ are on the same component. We start with the following key definition.

Definition. *For any top alignment point q, let $left_r(q)$ be the rightmost top component point p such that $minsep(p,q) = sep(r,succ(r))$, and be undefined if no such component point exists. (Note that there can be two p satisfying the equality if they are right and left component endpoints that coincide in the top minimum separation configuration.)*

Note that by Lemma 2.1, $left_r(q)$ is the leftmost point p for which $D^{p,q}_{r,succ(r)}$ is finite unless $left_r(q)$ is the left endpoint of a component. If $left_r(q)$ is $tl(i)$ for some $i > 1$, then $tr(i-1)$ is also legal, but $D^{tl(i),q}_{r,succ(r)} = D^{tr(i-1),q}_{r,succ(r)}$, since there are no terminals between r and $succ(r)$. Thus in our data structure representing $D^{*,*}_{r,succ(r)}$, $D^{left_r(q),q}_{r,succ(r)}$ will be at the head of the list for the element of $R^{\theta}_{r,succ(r)}$ that represents (or equals) q.

In computing $D^{*}_{succ(r)}$, we treat the members of $R^{\theta}_{succ(r)}$ from right to left in sequence, starting with $tl(c_t+1)$. For each component point q in the sequence, we start by computing $left_r(q)$, if it exists. Similarly, for each interval point representing an interval (q_L, q_R) in the sequence, we start by computing $left_r(q_L)$ and $left_r(q_R)$ (Using our data structure for $R^{\theta}_{succ(r)}$, the computations of $left_r(q)$ can be done in overall time $|R_{succ(r)}|$, since these values proceed from right to left roughly in parallel with the value of q, and we can keep track of them merely be sliding a second pointer through the list.) There are now seven cases to consider, depending on the identity of the point q in question and the location (or existence) of $left_r(q)$ or $left_r(q_L)$.

(1) q is a member of $R_{succ(r)}$ on a component and $left_r(q)$ is on the same component.
(2) q is a member of $T_t \cup E_t$ not satisfying (1).
(3) q is a member of $R_{succ(r)}$ not satisfying (1) or (2).
(4) q is an interval point representing (q_L, q_R) and $left_r(q_L)$ is on the same component as the interval.
(5) q is an interval point not satisfying (4) and q_R or $left_r(q_R)$ is a member of $T_t \cup E_t$.
(6) q is an interval point not satisfying (4) or (5).
(7) q is a gap point or $tr(0)$.

Note that we do not have to make a special case of our initial rightmost value of $q = tl(c_t+1)$, since that q is neatly covered by Case (3).

Now there are $O(N)$ members of $R^{\theta}_{succ(r)}$ meeting the requirements of (2), (5), and (7), and $O(|R_{succ(r)}|) = O(N^2)$ members satisfying each of (1), (3), (4) and (6). ALGORITHM 2 is designed so that it takes $O(N)$ time to compute $D^{q}_{succ(r)}$ in the former three cases, and constant time in the latter four. We shall illustrate this for cases (1) and (2).

In the analysis that follows, we use the following shorthands. For $tr(0) \le p \le q$, let $density^q_{succ(r)}(p) = \max(D^p_r, D^{p,q}_{r,succ(r)})$. Recall that by (5.1) our goal is to calculate $D^q_{succ(r)} = \min\{density^q_{succ(r)}(p) : tr(0) \le p \le q\}$. The following notation identifies a natural set that contains all ''legal'' candidates for p in the above minimization. For any top component point q, let $legal_r(q)$ be the set of top alignment points p such that $D^{p,q}_{r,succ(r)} < \infty$.

Case (1): If q and $left_r(q)$ are on the same component and it is not the case that $left_r(q) = tl(i)$ for some $i > 1$, then by Lemma 4.1, $legal_r(q) = \{left_r(q)\}$. Thus $D^q_{succ(r)} = density^q_{succ(r)}(left_r(q))$. If $left_r(q) = tl(i)$, then $legal_r(q) = \{left_r(q), tr(i-1)\}$. However, $D^{tl(i)}_r \le D^{tr(i-1)}_r$ and $D^{tl(i),q}_{r,succ(r)} = D^{tr(i-1),q}_{r,succ(r)}$. Thus, it again suffices to compute $density^q_{succ(r)}(left_r(q))$. The value $density^q_{succ(r)}(left_r(q)) = \max(D^{left_r(q)}_r, D^{left_r(q),q}_{r,succ(r)})$ can be computed in constant time as follows. Since we already have a pointer to $left_r(q)$, the first term can be found in constant time from our data structure for D^*_r. By Lemma 4.1, $D^{left_r(q),q}_{r,succ(r)}$ is equal to the leftmost finite value of $D^{*,q}_{r,succ(r)}$; so the value of the second term is available in constant time from our data structure for $D^{*,*}_{r,succ(r)}$.

Case (2): Suppose q is on component K^t_i. Since case (1) does not hold, $left_r(q)$ is either undefined or on K^t_j for some j, $1 \le j < i$. (In the former case, we shall say that $j = 0$.) The set $legal_r(q)$ thus consists of (a) all gap points $tr(h)+\Delta$, $j \le h < i$, (b) all points that lie on components K^t_h, $j < h < i$, and (c) $left_r(q)$ and all points to its right on K^t_j (when $j > 0$). The computation of $density^q_{succ(r)}(p)$ for all candidates satisfying (a) is $O(c_t) = O(N)$, since that is a bound on the number of such candidates, and our data structures allow us to access the relevant values of D^p_r and $D^{p,q}_{r,succ(r)}$ in constant time per value, given the augmentations we added for gap points. If $i = 1$, we are done, since no component points can be legal candidates for p. So assume $i > 1$ and such candidates exist. We handle them as follows:

Let $p_0, p_1, \ldots, p_s$ be the left-to-right sequence of members of $T_t \cup E_t$ that are legal candidates, with p_0 identified with $left_r(q)$ whether it is in $T_t \cup E_t$ or not, and p_s being $tr(i-1)$. p_0 is taken to be $tr(0)$ if $left_r(q)$ does not exist. Since we are processing points in $R_{succ(r)}$ from right to left, we can access p_s in $O(1)$ time by keeping a pointer to $tl(i)$ when we are processing q on component K^t_i. We can determine $density^q_{succ(r)}(p_h)$, $0 \le i \le s$ in overall time $O(s) = O(N)$, given the augmentations added to out data structures to handle the set $T_t \cup E_t$. The rest of the candidate set is divided into open intervals I_h, $0 \le h < s$, where I_h is the open interval lying between p_h and p_{h+1} if they are both on the same component, and is vacuous if p_h is a right endpoint. For each nonempty interval I_h, all members p give rise to the same value of $D^{p,q}_{r,succ(r)}$ (because the same set of terminals will lie in the interval $[r, succ(r)]$ in the corresponding placement). Thus to minimize $density^q_{succ(r)}(p)$ for such an interval, we need only find a p that minimizes D^p_r in the interval. But note that we have already precomputed both the identity of such a point and the minimum value. They are simply $pRMIN^{p_h}_r$ and $DRMIN^{p_h}_r$. (Note that p_0 may be represented by an interval point for $DRMIN^*_r$.) Thus the time to find the best p in a given interval is $O(1)$, and the time to find the intervals over all is $O(s) = O(N)$. Combining the cost of this computation with the earlier ones for the gap points and the p_h, we conclude that the overall time to find the member of $legal_r(q)$ that minimizes $density^q_{succ(r)}(p)$ is itself $O(N)$.

ALGORITHM 3. Assume without loss of generality that we have oriented the channel so that the number of components on the top of the channel is no more than the number on the bottom ($c_t \le c_b$). The above algorithm and its analysis make the worst case assumptions that c_t grows as $\Theta(N)$ and that the R^θ_v are $\Theta(N^2)$. In practice one might hope that both these values could be smaller. Furthermore, we are computing $D^*_{succ(v)}$ using each point in R_v, the set of *potential* points of change for D^*_v. Many of the points in R_v may not be true points of change, i.e. they may not represent a change of value for D^*_v. Thus we can further reduce the running time by restricting our calculations to set R^{true}_v, the true points of change for D^*_v, producing only true points of change for $D^*_{succ(v)}$. Our third algorithm has running time $O(N^2 c_t + \Sigma_{v \in E_b \cup T_b} |R^{true}_v|)$, to take advantage of these possibilities.

If we are to obtain the above running time bound, we can no longer afford to explicitly compute the functions $D^{*,*}_{v,succ(v)}$, since the descriptions of those functions can each be of size $\Theta(N^2)$, implying an $\Omega(N^3)$ lower bound. We thus compute additional auxiliary functions that enable us to derive $D^*_{succ(v)}$ from D^*_v directly, avoiding those parts of the computation $D^{*,*}_{v,succ(v)}$ that can be shown to be irrelevant. (Details are quite involved and must be postponed to the full paper.) Computing the auxiliary functions costs us $\Sigma_{v \in E_b \cup T_b} |R_v|$ overall and constructing the representations of the $D^*_{succ(v)}$ contributes and additional $N_b N_t c_t = O(N^2 c_t)$ overall. When all components are about the same size, a top component cannot simultaneously overlap more than two bottom components and each R_v has size $O(N_v N_t + N_{cmax} N_t) = O(N_{cmax} N)$, where N_{cmax} is the maximum number of terminals on any one component. Then, the optimum speedup comes when both N_{cmax} and c_t are $O(\sqrt{N})$, in which case the time becomes $O(N^{2.5})$.

Computing the Optimum Placement. The algorithms described so far need only $O(N^2)$ space, but if so restricted can return only the *value* of the optimum density. As is typical with dynamic programming algorithms, actually constructing a placement that realizes the optimum density requires significantly more space. In particular, one will need to store representations of all the functions D^*_v and also the pointers needed to reconstruct the computation path that led us to determine each function value. The worst-case bound for this is $O(N^3)$. (If space is more critical than time,

one can maintain the $O(N^2)$ space bound by spending $O(N^4)$ time overall doing recomputations.)

Alternatively, having found the minimum possible density, one might wish to generate a placement that optimizes some secondary criterion, subject to the constraint that the density be optimal. Building on the results presented here, Chao and LaPaugh [ChLa92], have devised an $O(N^3)$ time and space algorithm that, given a density target, will find a placement with minimum channel length among all those placements that have density less than or equal to the target, if any such placements exist. (This algorithm could be used to solve our original problem by binary search in time $O(N^3 \log N)$ but would probably be most useful in the indicated post-processing mode.)

References

[AtHa86] Atallah, M.J., Hambrusch, S.E., ''Optimal Rotation Problems in Channel Routing,'' *IEEE Transactions on Computers*, Vol C-35, No 9, (Sept. 1986) pp. 843-847.

[BBL84] Baker, B. S., S. N. Bhatt, and F. T. Leighton, ''An Approximation Algorithm for Manhattan Routing'', *Advances in Computing Research 2 (VLSI Theory)*, F. P. Preparata, ed., JAI Press, 1984, pp. 205-229.

[BrRi81] Brown, D.J., Rivest, R.L., ''New Lower Bounds for Channel Width,'' *VLSI Systems and Computations* (Proceedings of the CMU Conference on VLSI Systems and Computations), Kung, Sproull, and Steele, eds., Rockville, Maryland: Computer Science Press, 1981, pp. 178-185.

[BuPe83] Burstein, M., and Pelavin, R., ''Hierarchical Channel Router'', *Integration*, Vol. 1, No. 1 (March 1983), pp. 21-38.

[CaWo91] Cai, Y., Wong, D.F., ''Optimal Channel Pin Assignment'', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 11 (November 1991), pp. 1413-1424.

[ChLa92] Chao, L.F., LaPaugh, A. S., ''Finding All Minimal Shapes in a Routing Channel'', Princeton University Department of Computer Science technical report TR-384-92, Aug. 1992.

[Deu76] Deutsch, D. N., ''A Dogleg Channel Router'', *Proc. of the Thirteenth Design Automation Conf.*, 1976, pp. 425-433.

[FiRi82] Fiduccia, C. M., and Rivest, R. L., ''A Greedy Channel Router'', *Proc. of the Nineteenth Design Automation Conf.*, 1982, pp. 418-424.

[GrSh93] Greenberg, R., and Shih, J-D., ''Minimizing Channel Density with Movable Terminals'', *Third Gread Lakes Symposium on VLSI*, 1993, pp. 1-5.

[GCW83] Gopal, Coppersmith and Wong, ''Optimal wiring of Movable Terminals'' *IEEE Transactions on Computers*, Vol C-32, No 9, (1983) pp. 845-858.

[LaPi83] LaPaugh, A., Pinter, R., ''On Minimizing Channel Density by Lateral Shifting,'' *1983 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*, IEEE, September 1983, pp. 123-124.

[LaPi90] LaPaugh, A., Pinter, R., ''Channel Routing for Integrated Circuits (a survey)'', *Annual Review of Computer Science*, Vol. 4, Joseph Traub editor, (Annual Reviews, Inc., Palo Alto, Ca.) 1990.

[LePi83] Leiserson, C., Pinter, R., ''Optimal Placement for River Routing,'' *SIAM Journal on Computing*, Vol. 12, No. 3 (August 1993), pp. 47-462.

[Pe76] Persky, G., ''PRO - an Automatic String Placement Program for Polycell Layout'', *Proc. of the Thirteenth Design Automation Conf.*, 1976, pp. 417-424.

[Szy85] Szymanski, T.G., ''Dogleg Channel Routing is NP-Complete,'' *IEEE Trans. on Computer-Aided Design* Vol. CAD-4, No. 1 (Jan. 1985) pp. 31-41.