MIT/LCS/TM-206

OPTIMAL PLACEMENT FOR RIVER ROUTING

Charles E. Leiseron
Ron Y. Pinter

October 1981

Optimal Placement for River Routing

Charles E. Leiserson
Ron Y. Pinter

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139–1986

Abstract

Programs for integrated circuit layout typically have two phases: placement and routing. The router should produce as efficient a layout as possible, but of course the quality of the routing depends heavily on the quality of the placement. On the other hand, the placement procedure ideally should know the quality of a routing before it routes the wires. In this talk we present an optimal solution for a practical, common version of this placement and routing problem.

River routing is the problem of connecting in order a set of terminals $a_1, \ldots, a_n$ on a line to another set $b_1, \ldots, b_n$ across a rectangular channel. Since the terminals are located on modules, the modules must be placed relative to one another before routing. This placement problem arises frequently in design systems like *bristle-blocks* where *stretch lines* through a module can effectively break it into several *chunks,* each of which must be placed separately. In this talk we shall present *concise* necessary and sufficient conditions for wirability which are applied to reduce the optimal placement problem to the graph-theoretic *single-source-longest-paths* problem. By exploiting the special structure of graphs that arise from the placement problem for rectilinear wiring, an optimal solution may be determined in linear time.

Key words and phrases: analysis of algorithms, graph theory, longest paths, placement and routing, river routing, VLSI layout.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract No. N00014-80-C-0622.

1. Introduction

River routing is a special routing problem which arises often in the design of integrated circuits, and it has been shown to be optimally solvable in polynomial-time for many wiring models (see in particular [Tompa] and [Dolev *et al.*]). In this paper we demonstrate that the placement problem for river routing is also polynomial-time solvable.

The general character of the placement problem for river routing is illustrated in Figure 1. Two sets of terminals $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ are to be connected by wires across a rectangular channel so that wire i is routed from a_i to b_i. The terminals on each side of the channel are grouped into *chunks* which must be placed as a unit. The quality of a legal placement—one for which the channel can be routed—can be measured in terms of the dimensions of the channel. The *separation* is the vertical distance between the two lines of terminals, and the *spread* is the horizontal dimension of the channel.

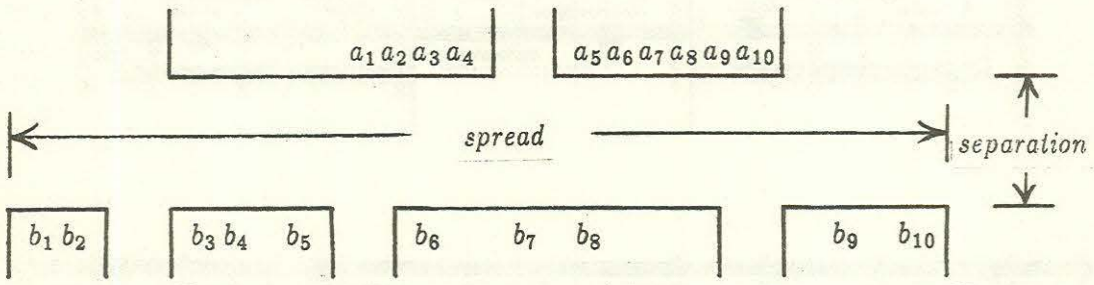


Figure 1: Two sets of chunks on either side of a rectangular channel. Terminal a_i must be connected to b_i for $i = 1, \ldots, 10$.

The *wiring model* gives the constraints that the routing must satisfy. Although our results can be generalized to include a variety of wiring models (see Section 5), we concentrate on the *(one-layer) square-grid model.* Crossovers are disallowed in the square-grid model, and all wires must take disjoint paths through the grid.

The placement problem for river routing arises often during ordinary integrated circuit design. A common instance is when the terminals of one or more modules are to be connected to drivers. The various independent "chunks" are the modules, which lie on one side of the channel, and the drivers, which lie on the other.

A more interesting manifestation of the placement problem occurs in the context of such design systems as bristle-blocks [Johannsen] and DPL/Daedalus [Batali *et al.*]. These systems encourage a designer to build plug-together modules so that the difficulties associated with general routing can be avoided. A designer may specify *stretch lines* which run through a module and allow the module to be expanded perpendicular to the stretch line, as demonstrated in Figure 2. When two independently designed modules are plugged together, stretch lines permit the terminals to be *pitch aligned,* that is, the distances between pairs of adjacent terminals are made to match the distances between their mates, and routing is avoided because the separation of the channel is zero. Unfortunately, this approach may not succeed unless stretch lines are put between every pair of adjacent terminals. The stretch lines may not only disrupt the internal structure of the modules, but the consequence may be an inordinate amount of stretching that leaves the channel with a large spread.

The other extreme is to forego stretching altogether and river route between the terminals. But the cost may still be large if a large separation is required in order to achieve a routing. A reasonable compromise is to place stretch lines where it is convenient, and then do a little stretching and a little routing. Determining how much of each to do is exactly the optimal placement problem for river routing.

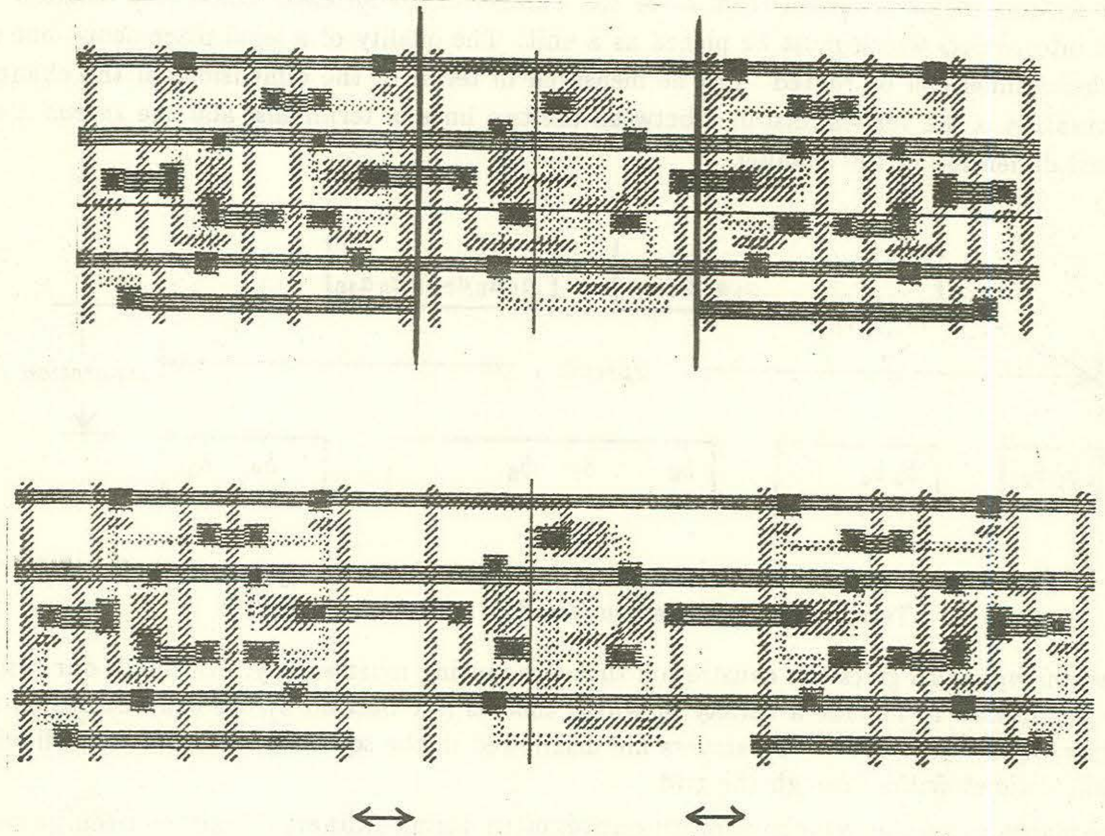


Figure 2: A module before and after stretching (courtesy of John Batali).

The remainder of this paper demonstrates that optimal solutions to the placement problem can be achieved efficiently. Section 2 gives a concise necessary and sufficient condition for a channel to be routable in the square-grid model. Section 3 shows that the form of this condition allows the placement problem to be reduced to the graph-theoretic problem of finding the longest paths from a source vertex to all other vertices in a graph. Based on this problem reduction, a linear-time algorithm for optimal placement is given in Section 4. Section 5 shows that the algorithm extends to wiring models other than the square-grid model, but its performance depends on the particular wirability conditions for the model. Section 6 discusses the application of our results to other routing situations and suggests further placement problems.

2. Necessary and Sufficient Conditions for Wirability

To demonstrate the results of this paper, we adopt an extremely simple wiring model: the *(one-layer) square-grid model.* All wires are constrained to run on an underlying grid of integer lattice points. The terminals $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ occupy grid points on opposite sides of the channel. No two wires may occupy a single grid point which enforces unit separation of wires. Figure 3 shows a solution to the problem of Figure 1 using this model.

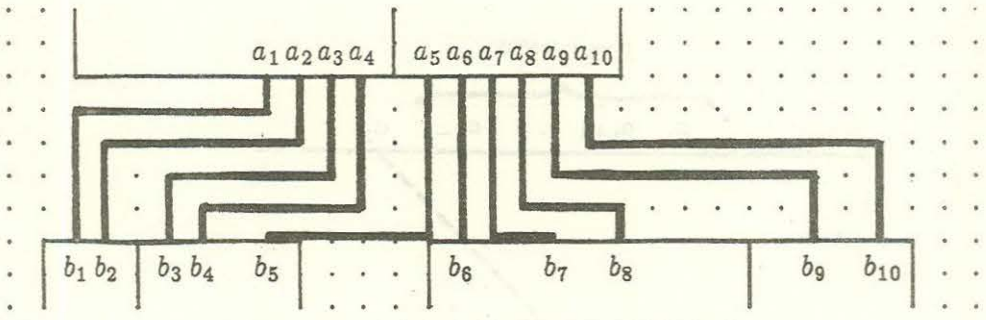


Figure 3: A possible solution to the problem in Figure 1 for which the separation is 5 and the spread is 27.

In order to establish constraints on wirability in this model, consider a straight line segment drawn from (x_1, y_1) to, but not including, (x_2, y_2). We ask the question, "How many wires can cross this line?" A simple analysis shows the answer is $\max(|x_2 - x_1|, |y_2 - y_1|)$. Without loss of generality, assume the situation is as in Figure 4, and look at the grid points immediately below the line, that is, $\{ (x, \lfloor y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \rfloor) \mid x = x_1, \ldots, x_2 - 1 \}$. Any wire crossing the line must perforce occupy one of these grid points, and therefore the number of such wires is bounded by the cardinality of this set.

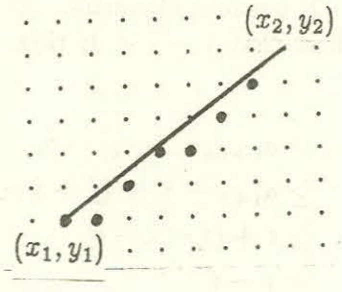


Figure 4: The number of wires crossing the half-open line segment is at most the number of grid points immediately below the line.

Let us now turn to the river routing problem and examine how this constraint can be brought to bear. Let $a_1, \ldots, a_n$ denote both the names of the terminals at the top of the channel and their x-coordinates, and let the same convention hold for the terminals $b_1, \ldots, b_n$ at the bottom of the channel. Consider a half-open line segment drawn from terminal a_j to terminal b_i as shown in Figure 5. The $j-i$ wires emanating from $a_i, \ldots, a_{j-1}$ must all cross this line, and similarly for a line drawn from b_j to a_i. In order for a channel with separation t to be routable, therefore, it must be the case that

$$\max(a_j - b_i, t) \geq j - i \quad \text{and} \quad \max(b_j - a_i, t) \geq j - i \tag{1}$$

for $1 \leq i \leq j \leq n$.

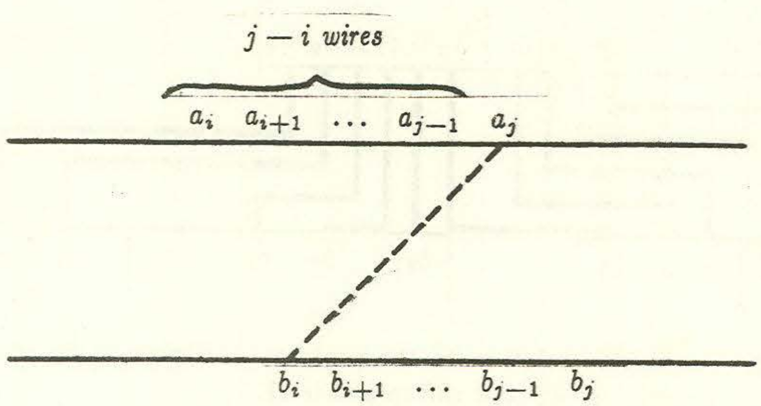

$j-i$ wires
a_i a_{i+1} ... a_{j-1} a_j
b_i b_{i+1} ... b_{j-1} b_j

Figure 5: The $j-i$ wires from $a_i, \ldots, a_{j-1}$ must cross the dashed line between b_i and a_j.

Although Condition (1) is a new condition for wirability, the analysis that leads to it is essentially the same as that in [Dolev *et al.*] and represents previous work in the field. A more compact condition exists, however, which is equivalent:

$$a_{i+t} - b_i \geq t \quad \text{and} \quad b_{i+t} - a_i \geq t \tag{2}$$

for $1 \leq i \leq n - t$. The channel is always routable if $t \geq n$.

Condition (1) implies Condition (2) because Condition (2) is a refinement of Condition (1). For the opposite direction, suppose first that $j - i < t$; then $\max(a_j - b_i, t) \geq t > j - i$. If $j - i \geq t$, on the other hand, then

$$\begin{aligned} a_j - b_i &= a_{i+t+(j-i-t)} - b_i \\ &\geq a_{i+t} - b_i + (j - i - t) \\ &\geq t + (j - i - t) \\ &= j - i \end{aligned}$$

since $a_{k+1} \geq a_k + 1$ for all $1 \leq k < n$. Thus the two conditions are indeed equivalent.

Figure 6 shows a simple geometric interpretation of Condition (2). The condition $a_{i+t} - b_i \geq t$ means that a line with unit slope going up and to the right from b_i must intersect the top of the channel at or to the left of terminal a_{i+t}. And if the condition fails, terminal b_j must be to the right of a_j for $i \leq j < i + t - 1$, that is, each wire from a a_j goes down and to the right, which can be shown to follow from the fact that $a_{j+1} \geq a_j + 1$. (For $b_{i+t} - a_i \geq t$ the line

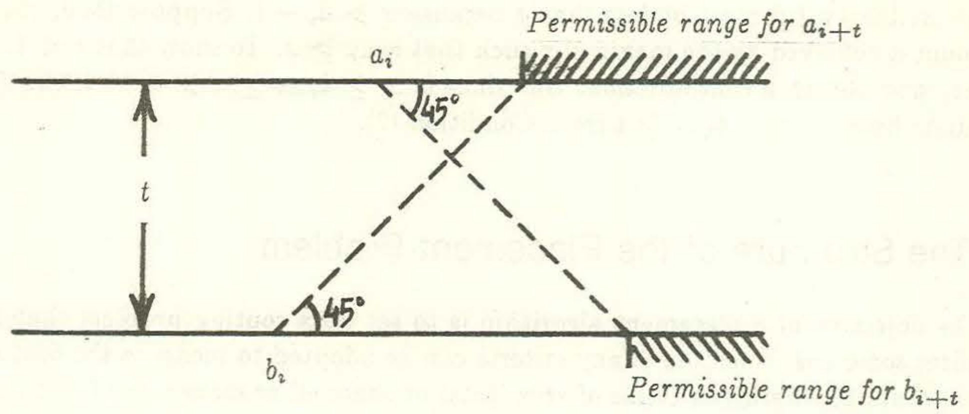

Permissible range for a_{i+t}
a_i
$45°$
t
$45°$
b_i
Permissible range for b_{i+t}

Figure 6: Geometric interpretation of $a_{i+t} \geq b_i + t$ and $b_{i+t} \geq a_i + t$.

with slope -1 going down and to the right from a_i must intersect the bottom of the channel at or to the left of terminal b_{i+t}.)

This geometric interpretation can be used to show that Condition (2) is not only a necessary condition for routability of the channel, but a sufficient condition as well. In fact, a simple greedy algorithm will successfully route a routable channel. Processing terminals left to right, the greedy algorithm routes each wire across the channel until it hits a previously routed wire; then it follows the contour of the opposite side until it reaches its destination.

To see that this algorithm works given Condition (2), we must be more precise about what paths are taken by the wires. Consider without loss of generality a block of consecutive wires that go down and to the right, that is, $a_i \leq b_i$ for all wires in the block. For any horizontal position x such that $a_i - t < x \leq b_i$, define

$$\eta_i(x) = \max(a_i - x, \max_{b_{i-r} \geq x} r).$$

The path of wire i is then described by the locus of points $\big(x + \eta_i(x), \eta_i(x)\big)$ for $a_i - t < x \leq b_i$.

A geometric interpretation of this formulation uses the same intuition as was given in Figure 6. The line with unit slope drawn from $(x, 0)$ where x is in the range $a_i - t < x \leq b_i$ must cross wire i. The value $\eta_i(x)$ gives the y-coordinate of wire i where it crosses this line of unit slope. The two-part maximum in the definition of $\eta_i(x)$ corresponds to whether the wire is being routed straight across the channel or whether it is following the contour of the bottom. The value of $\eta_i(x)$ for the latter situation is the number of wires to the left of wire i which must cross the line of unit slope.

We must now show that the locus of points for a wire is a path, that the paths are disjoint, and that they never leave the channel. That the locus of points is indeed a path can be seen by observing that as x ranges from $a_i - t$ to b_i, the initial point is $(a_i, t-1)$, the final point is $(b_i, 0)$, and with a change of one in x the coordinates of the path change by a single grid unit in exactly one of the two dimensions. To show that the paths are disjoint, consider two adjacent wires i and $i+1$, and observe for $a_{i+1} - t < x \leq b_i$ that $a_i - x < a_{i+1} - x$ and $\max_{b_{i-r} \geq x} r < \max_{b_{i+1-r} \geq x} r$, and therefore $\eta_i(x) < \eta_{i+1}(x)$.

To show a path of a wire never leaves the channel, we demonstrate that $\eta_i(x) < t$ for all i and x in the associated range. It is for this part of the proof that we need the assumption that

Condition (2) holds. If for a wire i, the two-part maximum in the definition of $\eta_i(x)$ is achieved by $a_i - x$, then $\eta_i(x)$ must be less than t because $x > a_i - t$. Suppose then, that the two-part maximum is achieved by the maximal r such that $b_{i-r} \geq x$. To show that $r < t$, we assume the contrary and obtain a contradiction. But since $b_{i-t} \geq b_{i-r} \geq x > a_i - t$, the contradiction is immediate because $a_i - b_{i-t} \geq t$ from Condition (2).

3. The Structure of the Placement Problem

The objective of a placement algorithm is to set up a routing problem that is solvable and minimizes some cost function. Many criteria can be adopted to measure the cost of a placement for river routing, whether in terms of area (total or channel) or some other function of spread and separation. A plot of minimal spread versus given separation reveals that the region of feasible placements may not be convex although the curve is guaranteed to be monotonically decreasing. (Figure 7 shows the plot for the problem of Figure 1.) Any measure of placement cost that is a function of spread and separation and which is monotonically increasing in each of spread and separation will therefore find a minimum on this curve.

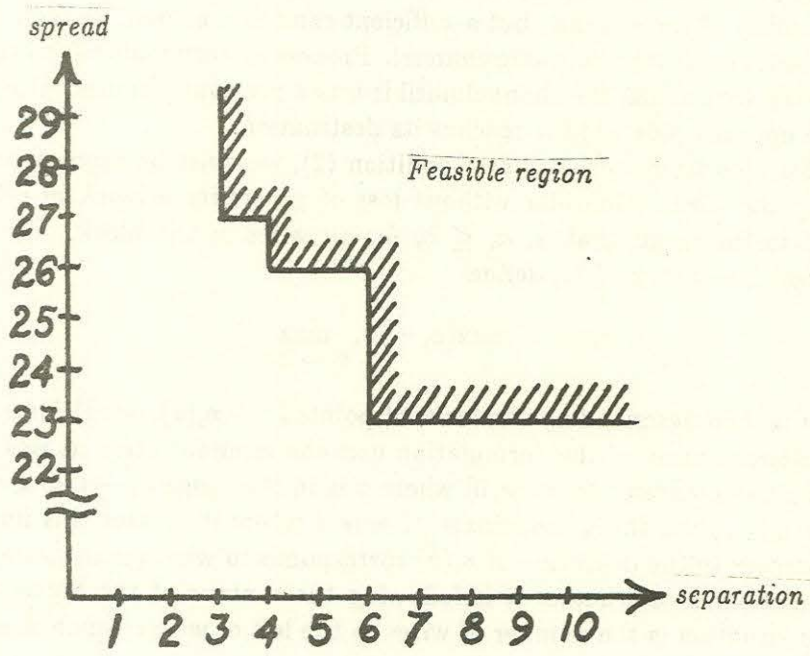
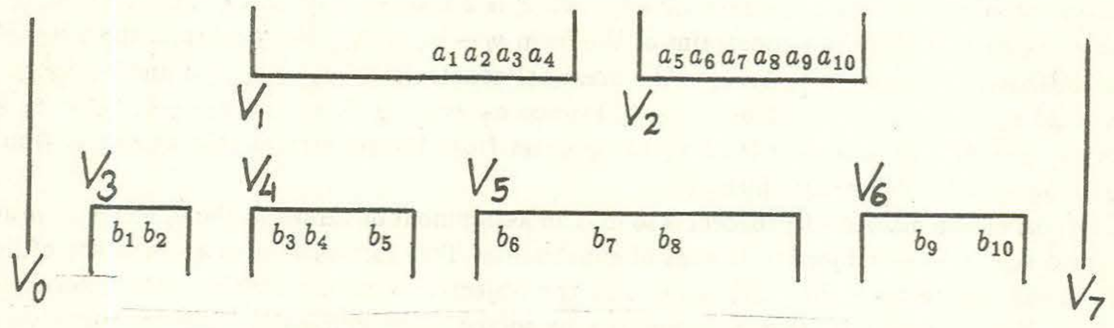


Figure 7: The curve of minimum spread versus separation for the example of Figure 1.

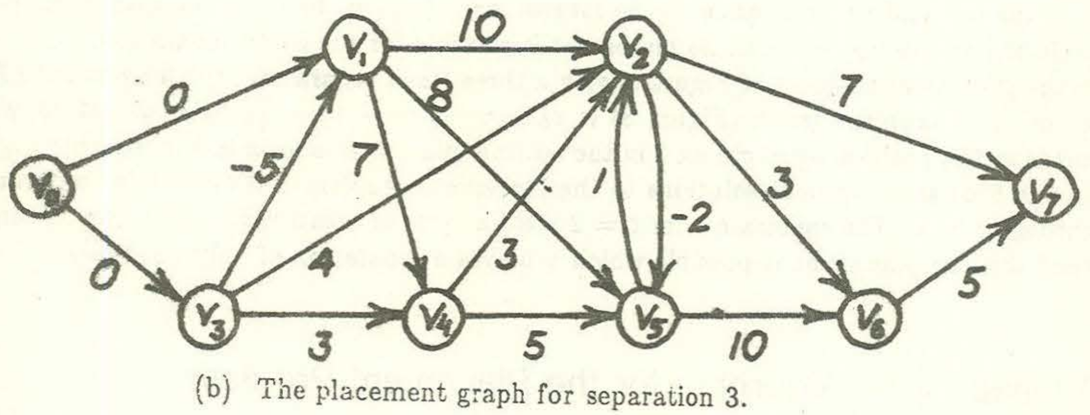
Thus we content ourselves with producing points on this curve, that is, *determining a placement which achieves the minimum spread for a given separation t,* if indeed the channel is routable in t tracks. If minimum separation is the goal, for example, binary search can determine the optimum t in $O(\lg t)$ steps. Since the algorithm presented in the next section determines a placement for fixed t in $O(n)$ time where n is the number of terminals, and since the separation need never be more than n, a minimum-separation placement can be achieved in $O(n \lg n)$ time.

For more general objective functions such as area, the optimum value can be determined in $O(n^2)$ time.

We now examine the character of the placement problem for river routing when the separation t is given. The n terminals are located on m chunks which are partitioned into two sets that form the top and bottom of the channel. For convenience, we shall number the chunks from one to k on the top, and $k+1$ to m on the bottom. The order of chunks on each side of channel is fixed, but they may be moved sideways so long as they do not overlap. For each chunk i, a variable v_i represents the horizontal position of its left edge. Any placement can therefore be specified by an assignment of values to these variables. We also add two variables v_0 and v_{m+1} to the set of variables, which represent the left and right boundaries of the channel. The spread is thus $v_{m+1} - v_0$. Figure 8(a) shows the eight variables for the example from Figure 1.



(a) Assignment of variables to chunks and channel boundaries.



(b) The placement graph for separation 3.

Figure 8: Representing the placement constraints as a graph for the example of Figure 1.

Since the relative positions of terminals within a chunk is fixed, the wirability constraints of Condition (2) can be reexpressed in terms of the chunks themselves to give placement constraints that any assignment of values to the v_i must satisfy. If terminal a_{i+t} lies on chunk h, and terminal b_i lies on chunk j, the constraint $a_{i+t} - b_i \geq t$ can be rewritten as $v_h - v_j \geq r_{hj}$, where

r_{hj} reflects t and the offsets of the terminals from the left edge of their respective chunks. The constraint between two chunks determined in this way will be the maximal constraint induced by pairs of terminals.

Additional constraints arise from the relative positions of chunks on either side of the channel. For each pair of adjacent chunks i and $i+1$, the constraint $v_{i+1} - v_i \geq w_i$ must be added to the set of placement constraints, where w_i is the width of chunk i. Four more constraints are needed which involve the boundary variables v_0 and v_{m+1}. For chunks 1 and $k+1$ which are leftmost on the top and bottom, the constraints $v_1 - v_0 \geq 0$ and $v_{k+1} - v_0 \geq 0$ enforce that these chunks lie to the right of the left boundary of the channel. For chunks k and m which are rightmost on the top and bottom, the relations $v_{m+1} - v_k \geq w_k$ and $v_{m+1} - v_m \geq w_m$ constrain them to lie to the left of the right boundary, where w_k and w_m are the widths of the chunks.

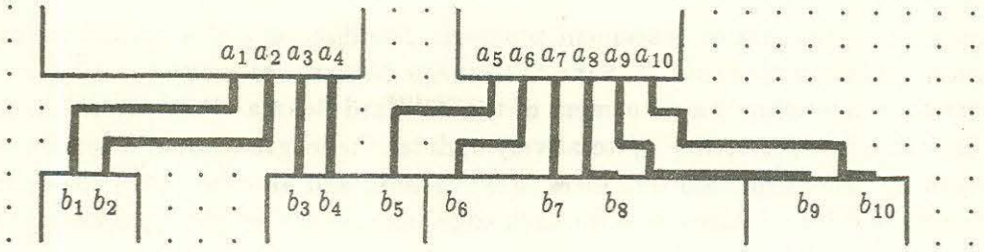
Figure 8(b) shows a *placement graph* which represents the constraints between chunks for the placement problem of Figure 1 where the separation is 3 tracks. A directed edge with weight δ_{kl} goes from v_k to v_l if there is a constraint of the form $v_l - v_k \geq \delta_{kl}$. For example, the weight of 1 on the *cross edge* going from v_5 to v_2 is the maximal constraint of $a_9 - b_6 \geq 3$ and $a_{10} - b_7 \geq 3$ which yield $v_2 - v_5 \geq -1$ and $v_2 - v_5 \geq 1$ since $a_9 = v_2 + 5$, $a_{10} = v_2 + 6$, $b_6 = v_5 + 1$, and $b_7 = v_5 + 4$. The *side edge* from v_4 to v_5 arises from the constraint that chunk 4, which is 5 units long, must not overlap chunk 5.

The goal of the placement problem is to find an assignment of values to the v_i which minimizes the spread $v_{m+1} - v_0$ subject to the set of constraints. This formulation is an instance of linear programming where both the constraints and the objective function involve only differences of variables. Not surprisingly, this problem can be solved more efficiently than by using general linear programming techniques. In fact, it reduces to a *single-source-longest-paths* problem in the placement graph. The length of a longest path from v_0 to v_{m+1} corresponds to the smallest spread of the channel that complies with all the constraints. The placement of each chunk i relative to the left end of the channel is the longest path from v_0 to v_i. If the placement graph has a cycle of positive weight, then no placement is possible for the given separation.

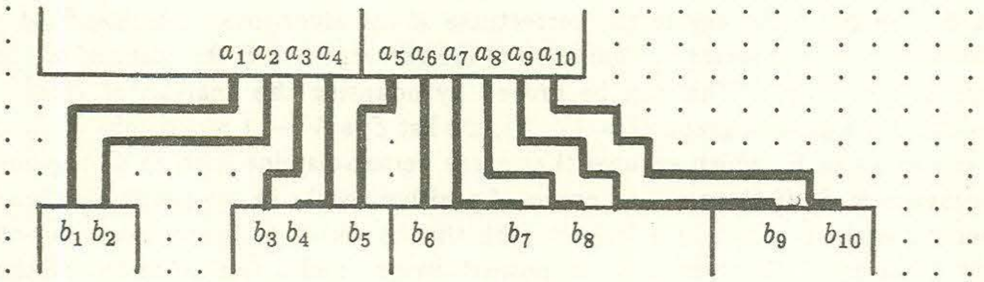
For the placement problem of Figure 1 with a three-track separation, the longest path from v_0 to v_2 in the placement graph (Figure 8) is $v_0 - v_1 - v_4 - v_5 - v_2$ with weight 13 which corresponds to the positioning of chunk 2 in the optimal placement shown in Figure 9(a). Figures 9(b) through 9(d) show optimal solutions to the placement problem of Figure 1 for separations $t = 4$ through $t = 6$. The constraints for $t = 2$ yield a cycle of positive weight in the placement graph, and thus no placement is possible which achieves a separation of only two tracks.

4. A Linear-Time Algorithm for the Placement Problem

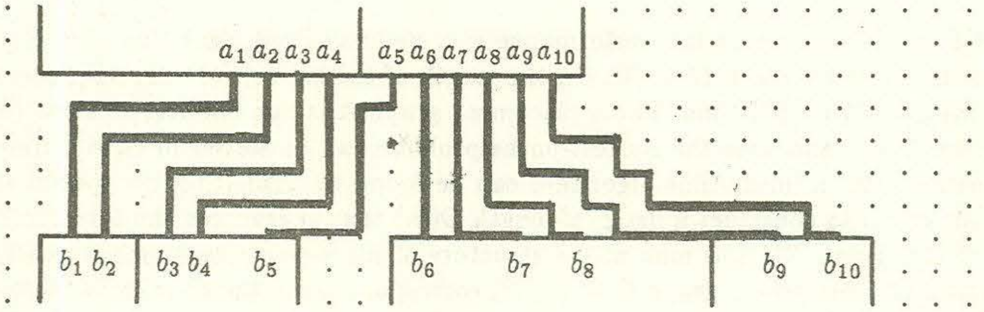
The analysis of Section 3 showed that the optimal placement problem for fixed-separation river routing was reducible to the single-source-longest-paths problem on a placement graph. For a general graph $G = (V, E)$ this problem can be solved in time $O(|V| \cdot |E|)$ by a *Bellman-Ford algorithm* [Lawler]. Better performance is possible, however, due to the special structure of placement graphs. This section reviews the Bellman-Ford algorithm, and shows how it can be adapted to give an $O(m)$-time algorithm for the longest-paths problem on a placement graph, where m is the number of chunks. Since the placement constraints can be generated in $O(n)$ time, where n is the number of terminal pairs, this algorithm leads to an optimal linear-time



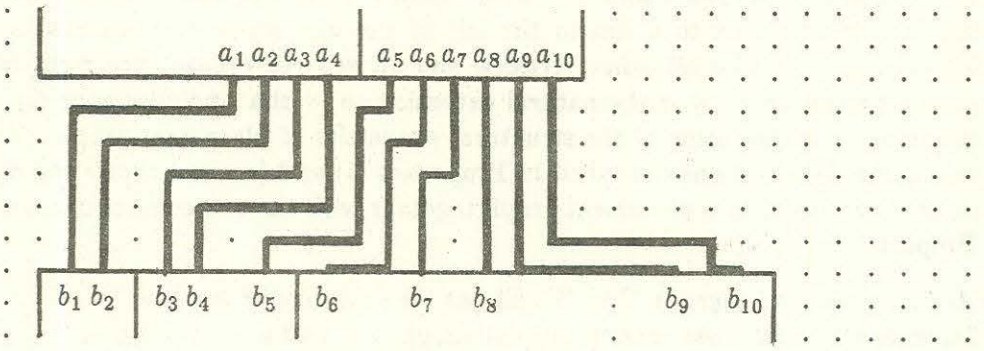
(a) Separation 3, spread 27.



(b) Separation 4, spread 26.



(c) Separation 5, spread 26.



(d) Separation 6, spread 23.

Figure 9: Optimal placements and routings for the problem of Figure 1 with separations ranging from $t = 3$ to $t = 6$.

algorithm for the fixed-separation placement problem. The discovery of a linear-time algorithm represents joint research with James B Saxe of Carnegie-Mellon University.

The linear-time algorithm is a refinement of the standard Bellman-Ford algorithm which for each vertex v_i where $i = 1, \ldots, m+1$, iteratively updates the length $\lambda(v_i)$ of a tentative longest path from v_0 to v_i. The algorithm initializes $\lambda(v_0)$ to zero, and all other $\lambda(v_i)$ to $-\infty$; then it sequences through a list $\mathcal{E}$ of edges, and for each edge (v_i, v_j) with weight δ_{ij} updates $\lambda(v_j)$ by

$$\lambda(v_j) \leftarrow \max(\lambda(v_j), \delta_{ij} + \lambda(v_i)).$$

The list $\mathcal{E}$ of edges is the key to the correctness of the algorithm. *The length of a longest path from the source v_0 to a vertex v_j converges to the correct value if the edges of the path form a subsequence of the list $\mathcal{E}$.* (This can be proved by adapting the analysis of [Yen].) In the normal algorithm for a general graph $G = (V, E)$, the list $\mathcal{E}$ is $|V| - 1$ repetitions of an arbitrary ordering of the edges in E, which ensures that every vertex-disjoint path in G beginning with v_0 is a subsequence of $\mathcal{E}$. If there are no cycles of positive weight in the graph G, then from v_0 to each other vertex in G, there is a longest path that is vertex-disjoint; hence the algorithm is guaranteed to succeed. The condition of positive-weight cycles can be tested at the end of the algorithm either by checking whether all constraints are satisfied or by simply running the algorithm through the edges in E one additional time and testing whether the values of any $\lambda(v_i)$ change.

The list $\mathcal{E}$ is also the key to the performance of a Bellman-Ford algorithm. For the general algorithm on an arbitrary graph $G = (V, E)$, the length of the list is $(|V| - 1) \cdot |E|$, and thus the algorithm runs in $O(|V| \cdot |E|)$ time. For a placement graph it is not difficult to show that both $|V|$ and $|E|$ are $O(m)$, and thus the longest-paths problem can be solved in $O(m^2)$ time by the general algorithm. But a linear-time algorithm can be found by exploiting the special structure of a placement graph to construct a list $\mathcal{E}$ of length $O(m)$ that guarantees the correctness of the Bellman-Ford algorithm. We now look at the structure of placement graphs more closely.

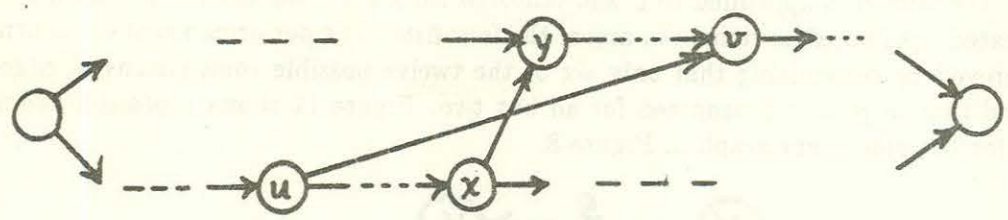
The vertices of a placement graph $G = (V, E)$ corresponding to the chunks on the top of the channel have a natural linear order imposed by the left-to-right order of the chunks. We define the partial order $\prec$ as the union of this linear order with the similar linear order of bottom vertices. Thus $u \prec v$ for vertices u and v if their chunks lie on the same side of the channel and the chunk that corresponds to u lies to the left of the one which corresponds to v. The left-boundary vertex v_0 precedes all other vertices, and all vertices precede the right-boundary vertex v_{m+1}. The partial order $\preceq$ is the natural extension to $\prec$ that includes equality.

The next lemma describes some of the structural properties of placement graphs. Figure 10 illustrates the impossible situations described in Properties (i) and (ii) and shows the only kind of simple cycle that can occur in a placement graph together with the two consecutive cross edges that satisfy Property (iii).

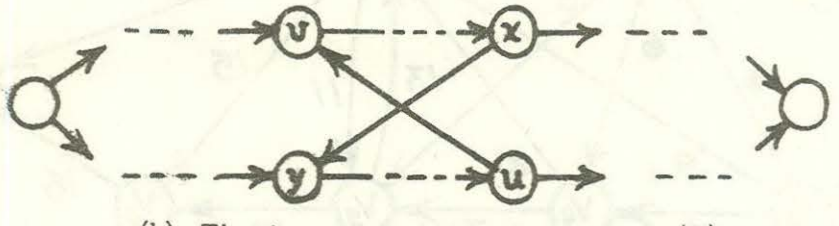
Lemma 1. *Any placement graph $G = (V, E)$ has the following properties:*

(i) *There do not exist cross edges (u, v) and (x, y) such that $u \prec x$ and $y \prec v$.*
(ii) *There do not exist cross edges (u, v) and (x, y) such that $v \prec x$ and $y \prec u$.*
(iii) *All cycles have two consecutive cross edges (u, v) and (v, w) such that $w \preceq u$.*

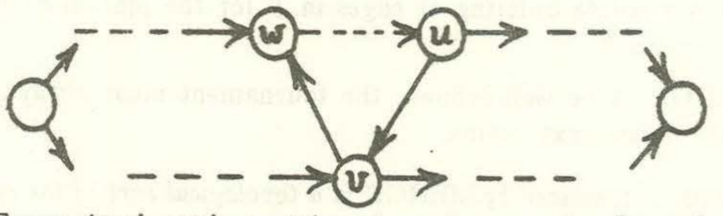
Proof. Properties (i) and (ii) can be proved by considering which of the terminal constraints from Condition (2) induce the edges in the placement graph. For each of these cases, suppose the edge (u, v) was caused by the terminals i in u and $i + t$ in v, and the edge (x, y) came from the



(a) The situation forbidden by Property (*i*).



(b) The situation forbidden by Property (*ii*).



(c) Every simple cycle contains at most one vertex from the top or at most one vertex from the bottom. The edges incident on the vertex are a consequence of Property (*iii*).

Figure 10: The properties of the placement graph enumerated in Lemma 1.

terminals j in x and $j+t$ in y. For Property (*i*) we have $u \prec x$ and $y \prec v$, and thus $i < j$ and $j + t < i + t$. Canceling t from this latter inequality obtains the contradiction. The assumption to be proved impossible in (*ii*) is that $v \prec x$ and $y \prec u$, which implies $i + t < j$ and $j + t < i$. Since t is nonnegative, we gain a contradiction.

To prove Property (*iii*), we need only consider simple (vertex-disjoint) cycles. Since no cycle can consist solely of side edges, every simple cycle must have a cross edge (u, v) going from bottom to top. In order to complete the cycle, there must be a top-to-bottom edge (w, x) such that $v \preceq w$ and $x \preceq u$. If $v = w$ or $x = u$, then the pair of edges satisfies Property (*iii*). But if $v \neq w$ and $x \neq u$, then the pair of edges violates Property (*ii*). ∎

Each edge in the placement graph is either a top edge, a top-bottom edge, a bottom-top edge, or a bottom edge. For each of these four sets of edges, there is a natural linear order of edges based on $\preceq$, where (u, v) precedes (x, y) for two edges in the same set if $u \preceq x$ and $v \preceq y$. Property (*ii*) guarantees that the linear order holds for two cross edges in the same set. Let TT, TB, BT, and BB be the four lists of edges according to the natural linear order, and include the two edges out of v_0 and the two edges into v_{m+1} in either TT or BB as appropriate.

The list $\mathcal{E}$ used by the Bellman-Ford algorithm is constructed by a merge of the four lists which we call MERGE. At each step of MERGE, a tournament is played among the first elements of each list. If (u, v) and (v, w) are the first elements of two lists, then (u, v) beats (v, w) if $w \not\preceq u$. Since there may be more than one edge beaten by none of the other three, ties are broken arbitrarily. The winner is appended to $\mathcal{E}$ and removed from the head of its list. The tournament is then repeated until no edges remain in any of the four lists. The performance of the tournament can be improved by recognizing that only six of the twelve possible comparisons of edges need be tried, and that $w \not\preceq u$ is guaranteed for all but two. Figure 11 shows a possible ordering of edges in $\mathcal{E}$ for the placement graph in Figure 8.

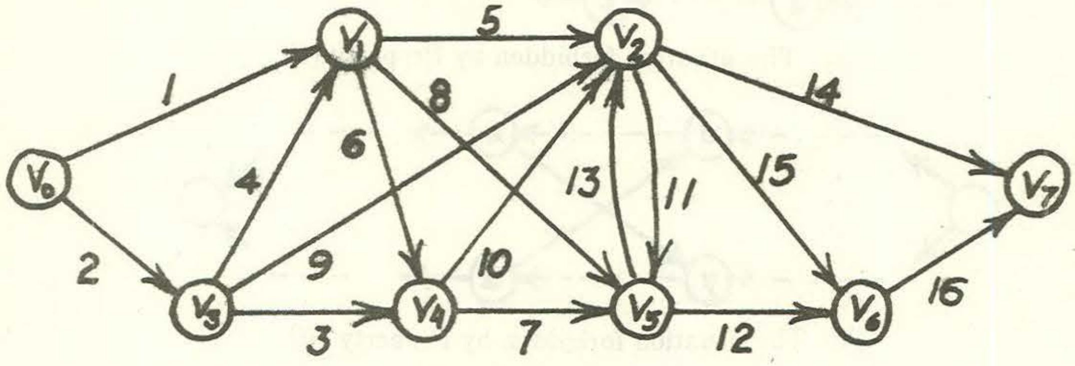


Figure 11: A possible ordering of edges in $\mathcal{E}$ for the placement graph in Figure 8.

In order for MERGE to be well-defined, the tournament must always produce a winner, which is a consequence of the next lemma.

Lemma 2. *The list $\mathcal{E}$ produced by* MERGE *is a topological sort of the edges of E according to the relation R where $(u,v)R(v,w)$ if $w \not\preceq u$.*

Proof. First, we show that the relation R is acyclic so that the edges can indeed be topologically sorted. By definition of R, a cycle in R induces a cycle in the placement graph. According to Property (*iii*), the cycle must have two consecutive cross edges (u,v) and (v,w) such that $w \preceq u$. But since $(u,v)R(v,w)$, we also have that $w \not\preceq u$, which is a contradiction.

The proof that MERGE topologically sorts the edges of E according to R makes use of the fact that if a vertex v is the tail of an arbitrary edge in any one of the four lists TT, TB, BT or BB, then for every $u \preceq v$ there is an edge in the same list emanating from u. Suppose that MERGE does not topologically sort the edges of E according to R. Then there is a first edge (u,v) in $\mathcal{E}$ such that there exists an edge (v,w) earlier in $\mathcal{E}$ and $(u,v)R(v,w)$. Consider the edge (x,y) in the same list as (u,v) that competed with (v,w) when (v,w) was the winner of the tournament. For each of the possible combinations of lists for (u,v) and (v,w), it can always be deduced that there is an edge emanating from y such that which makes (x,y) an earlier violator of the topological sort than (u,v). ∎

Since each edge of E is included exactly once in the list $\mathcal{E}$ created by MERGE, the Bellman-Ford algorithm applied to $\mathcal{E}$ has a running time linear in the number of chunks. The correct values for longest paths are produced by the algorithm if for every vertex v, there is a subsequence of $\mathcal{E}$ that realizes a longest path from v_0 to v, under the assumption that there are no positive-weight cycles in the placement graph. Since for every longest path, there is a vertex-disjoint longest path, the following theorem proves the correctness of this linear-time Bellman-Ford algorithm.

Theorem 3. *Let G be a placement graph with left-boundary vertex v_0. Then every vertex-disjoint path beginning with v_0 is a subsequence of the list $\mathcal{E}$ created by the procedure* MERGE.

Proof. We need only show that every pair of consecutive edges in a vertex-disjoint path from v_0 satisfies R because then Lemma 2 guarantees that the path is a subsequence of $\mathcal{E}$. Suppose (u,v) and (v,w) are two consecutive edges on a vertex-disjoint path from v_0 which violate R, that is, $w \preceq u$. If either (u,v) or (v,w) is a side edge, the pair must satisfy R, and thus both must be cross edges with the vertices u and w on the same side. Since if $w = u$, the path is not vertex-disjoint, we need only show that $w \prec u$ is impossible.

Assume, therefore, that $w \prec u$, and consider the initial portion of the path from v_0 to u. Since $v_0 \prec v$ and $v_0 \prec w$, there must be an edge (x,y) on the path which goes from the set of vertices to the left of (v,w) to the right of (v,w) in order to get to u. But then either Property (i) or Property (ii) is violated depending on whether $x \prec v$ and $w \prec y$, or $y \prec v$ and $x \prec w$. ∎

5. Other Wiring Models

The reduction from the fixed-separation placement problem in the square-grid model to the single-source-longest-paths problem is possible because the wirability constraints can all be written in the form $v_i - v_j \geq \delta_{ij}$. Thus for any wiring model where wiring constraints can be written in this form, the reduction will succeed. Also, it should be observed that in general, the performance of the single-source-longest-path algorithm will not be linear, but will be a function of the number of constraints times the number of variables. This section reviews other models and gives the necessary and sufficient wirability constraints for each.

1. *One-layer, gridless rectilinear* ([Dolev *et al.*]). Wires in this model must run horizontally or vertically, and although they need not run on grid points, no two wires can come within one unit of each other. The wirability constraints for this model are the same as for the square grid model:

$$a_{i+t} - b_i \geq t \quad \text{and} \quad b_{i+t} - a_i \geq t$$

for $1 \leq i \leq n-t$. As with the square-grid model, the fixed-separation placement algorithm for this model can be made to run in linear time.

2. *One-layer, gridless, rectilinear and forty-five degree* ([Dolev and Siegel]). This model is the same as the gridless rectilinear, but in addition wires can run which have slope ± 1. The constraints in this case are

$$a_{i+r} - b_i \geq r\sqrt{2} - t \quad \text{and} \quad b_{i+r} - a_i \geq r\sqrt{2} - t$$

for $t/\sqrt{2} \leq r \leq t$ and $1 \leq i \leq n-r$. The placement algorithm for this model runs in $O(\min(tm^2 + tn, m^3 + tn, m^3 + n^2))$ time.

3. *One-layer, gridless* ([Tompa]). Wires can travel any direction. The constraints are

$$a_{i+r} - b_i \geq \sqrt{r^2 - t^2} \quad \text{and} \quad b_{i+r} - a_i \geq \sqrt{r^2 - t^2}$$

for $t \leq r \leq n$ and $1 \leq i \leq n-r$. The placement algorithm runs in $O(m^3 + n^2)$ time.

4. *Multilayer models.* All the models presented until now have been one-layer models. It is natural to generalize to l-layer models in which wires may travel on different layers. Remarkably, optimal routability can always be achieved with no contact cuts ([Baratz]), that is, a wire need never switch layers. The necessary and sufficient conditions for these multilayer models are a natural extension of the one-layer conditions. For example, in the one-layer, gridless, rectilinear model the conditions are modified for l layers to be

$$a_{i+lt} - b_i \geq t \quad \text{and} \quad b_{i+lt} - a_i \geq t$$

for $1 \leq i \leq n - lt$.

There are some wiring models, however, where upper and lower bounds for wirability do not meet. For these models a constraint graph which represents upper bounds will give the best possible placement for those bounds. A graph representing lower bounds will give lower bounds on the best possible placement. Together, bounds can be established for some of these models, and heuristic algorithms invoked to attempt routing within the feasible range of optimality.

6. Extensions and Conclusions

A variety of related placement problems can be solved by the method described in this paper. Some entail extensions to the problem specifications, others employ different wiring models. In this section we shall mention a few extensions we can handle and suggest further research on more complicated problems.

- *Nonriver routing.* The placement algorithm gives optimal placements for river routing, but there are other routing configurations for which it works optimally as well. One example is the two-layer, any-to-any routing problem where two sets of terminals must be connected across a channel, but they may be connected in any order.

- *Parallel Channels.* Multiple, parallel horizontal channels are easily handled within the same graph-theoretic framework. Each row of chunks is represented by a chain of vertices (from a common left margin to a common right margin), and the wiring conditions in the channels are represented by edges linking adjacent chains. The optimal placement is achieved by solving the whole system, using the Bellman-Ford algorithm. We do not know how to obtain (or show the impossibility of) a linear-time algorithm to do the job in general, in contrast to the special case (one channel) discussed in Section 4. However, the number of edges in the graph is still linear in the number of vertices (chunks), thus the Bellman-Ford algorithm runs in time $O(n + m^2)$ (where n is the number of nets, m — the number of chunks).

- *Range-Terminals.* In some routing situations terminals occupy not a single point, but rather a contiguous region along the edge of the channel. For example, the terminal might be a wire that runs along the edge of the chunk, and connection can be made to the wire anywhere. The additional flexibility of viewing a terminal as a contiguous range of points can be exploited by both the greedy routing algorithm and the placement algorithm in any of the river-routing models we have discussed.

Each range-terminal is specified by an interval $[a_i^L, a_i^R]$ or $[b_i^L, b_i^R]$. The greedy routing algorithm operates as before with minor changes. If the range-terminals overlap, the wire is routed straight across. Otherwise, assume without loss of generality that $a_i^R < b_i^L$, and use the standard greedy algorithm to route a wire from a_i^R to b_i^L.

The wirability conditions for placement are accordingly adjusted. In the rectilinear case, for example, the condition $a_{i+t} - b_i \geq t$ is rewritten as $a^R_{i+t} - b^L_i \geq t$ and condition $b_{i+t} - a_i \geq t$ becomes $b^R_{i+t} - a^L_i \geq t$. The transformation to chunk variables is as before and the placement algorithm is unchanged.

An interesting extension of the river-routing problem studied here is the two-dimensional problem illustrated in Figure 12. In the figure, a line between two chunks indicates that wires must be routed between them. Unfortunately, in order to optimally solve this general problem, it appears that the constraints indicated by the lines must be convex in both dimensions, not just in one as is the case for the wiring models considered here. When the constraints are convex, however, convex programming can be used to optimize a cost function such as the area of the bounding box of the layout. One model which gives convex constraints for the general two-dimensional problem is the one in which all wires must be routed as straight line segments between terminals such that no minimum spacing rules are violated. This model is not particularly interesting from a practical standpoint, however. Heuristics for solving the related two-dimensional *compaction* problem by repeatedly compacting in one dimension and then the other can be found in [Hsueh].

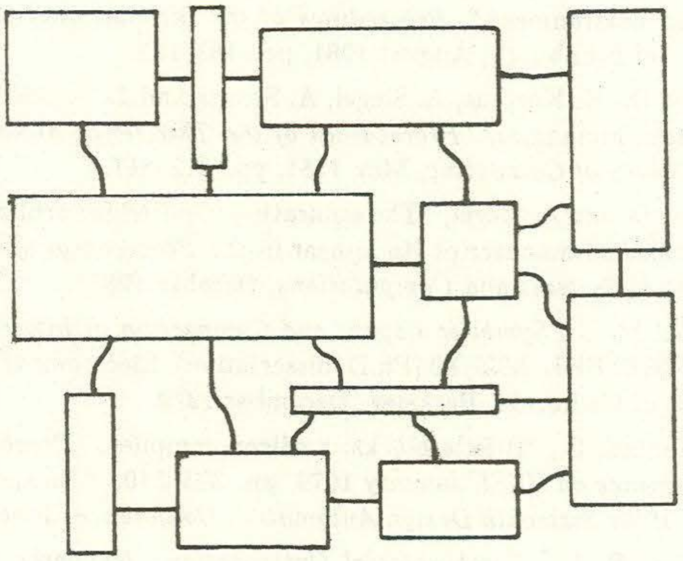


Figure 12: A two-dimensional extension to the river-routing problem. A solid line between two modules indicates routing occurs between them.

A major deficiency of placement algorithms is that they lack knowledge about the wirability of the routing problems that they set up. We have shown for river routing that wirability conditions can be translated directly into placement constraints without the overhead of wiring the channel. For rectilinear river routing, the running time of the greedy wiring algorithm is $O(n^2)$, and any cost function for placement that is monotonic in spread and separation can be optimized in $O(n^2)$ time without the overhead of routing. Studying wirability in the general case may lead to the development of heuristics for wirability that do not involve routing. A program that uses this heuristic knowledge should be able to outperform the iterative place-route, place-route programs that dominate today.

Acknowledgments. We would like to thank Howie Shrobe of the MIT Artificial Intelligence Laboratory for posting the plots of the data paths from the Scheme81 chip which inspired our interest in this placement problem and for his valuable comments on the practicality of our work. We would also like to thank Alan Baratz and Ron Rivest from the MIT Laboratory for Computer Science for numerous helpful discussions, and Shlomit Pinter also from the Laboratory for Computer Science for influencing the direction of our proof of Theorem 3. Finally, special thanks to Jim Saxe of Carnegie-Mellon University for his key contributions to the linear-time algorithm for longest-paths.

References

[Baratz] Baratz, A. E., *Algorithms for Integrated Circuit Signal Routing* (Ph.D. dissertation), Dept. of Electrical Engineering and Computer Science, M.I.T., August 1981.

[Batali *et al.*] Batali, J., N. Mayle, H. Shrobe, G. Sussman and D. Weise, "The DPL/ Daedalus design environment," *Proceedings of the International Conference on VLSI,* Univ. of Edinburgh, August 1981, pp. 183–192.

[Dolev *et al.*] Dolev, D., K. Karplus, A. Siegel, A. Strong and J. D. Ullman, "Optimal wiring between rectangles," *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing,* May 1981, pp. 312–317.

[Dolev and Siegel] Dolev, D. and A. Siegel, "The separation required for arbitrary wiring barriers," unpublished manuscript (to appear in the *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981).

[Hsueh] Hsueh, M.-Y., *Symbolic Layout and Compaction of Integrated Circuits,* Memo No. UCB/ERL-M79/80 (Ph.D. dissertation), Electronics Research Laboratory, Univ. of California, Berkeley, December 1979.

[Johannsen] Johannsen, D., "Bristle blocks: a silicon compiler," *Proceedings of the Caltech Conference on VLSI,* January 1979, pp. 303–310. Also appears in the *Proceedings of the Sixteenth Design Automation Conference,* June 1979, pp. 310–313.

[Lawler] Lawler, E. L., *Combinatorial Optimization: Networks and Matroids,* Holt, Rinehart and Winston, New York, 1976.

[Tompa] Tompa, M., "An optimal solution to a wire-routing problem," *Proceedings of the Twelfth Annual Symposium on Theory of Computing,* April–May 1980, pp. 161–176.

[Yen] Yen, J. Y., "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics,* Vol. 27, No. 4, July 1970, pp. 526–530.