Algorithmica (1987) 2: 113–129

Algorithmica
© 1987 Springer-Verlag New York Inc.

Global Wire Routing in Two-Dimensional Arrays[1]

R. M. Karp,[2] F. T. Leighton,[3] R. L. Rivest,[3] C. D. Thompson,[2]
U. V. Vazirani,[2] and V. V. Vazirani[4]

Abstract. We examine the problem of routing wires of a VLSI chip, where the pins to be connected are arranged in a regular rectangular array. We obtain tight bounds for the worst-case "channel-width" needed to route an $n \times n$ array, and develop provably good heuristics for the general case. Single-turn routings are proved to be near-optimal in the worst-case.

A central result of our paper is a "rounding algorithm" for obtaining integral approximations to solutions of linear equations. Given a matrix $\mathbf{A}$ and a real vector $\mathbf{x}$, then we can find an integral $\hat{\mathbf{x}}$ such that for all i, $|\hat{\mathbf{x}}_i - \mathbf{x}_i| < 1$ and $(\mathbf{A}\hat{\mathbf{x}})_i - (\mathbf{A}\mathbf{x})_i < \Delta$. Our error bound Δ is defined in terms of sign-segregated column sums of $\mathbf{A}$:

$$\Delta = \max_j \left(\max \left\{ \sum_{i:\mathbf{a}_{ij}>0} \mathbf{a}_{ij}, \sum_{i:\mathbf{a}_{ij}<0} -\mathbf{a}_{ij} \right\} \right).$$

Key Words. Global routing, Gate arrays, Integer programming, Linear programming, Computer-aided design for integrated circuits.

1. Problem Definition. We use a classical model of a gate-array wherein the chip area is considered to be divided into a uniform $n \times n$ array of square cells. Each cell contains p pins (connection points for logic elements). Each instance of our routing problem specifies a collection of nets where each net is specified as a set of pins. (Each pin is on at most one net.) Each net is to be connected together by horizontal and vertical wires. Unless stated otherwise, we assume that $p = 1$ and that each net connects exactly two pins.

A placement (planar embedding) of the underlying circuit is implicit in an instance of the gate-array routing problem. The only remaining work is to route the wires between the pins. For this reason, the gate-array routing problem is a special case of (and perhaps easier than) the general placement and routing problem studied in [T], [L3], [L1], [L2], [BL], and [CR5].

It is common to solve a gate-array routing problem instance P in two steps:

1. Compute a global routing R specifying for each net the set of cells and cell edges to be traversed by the wiring for that net.

[1] This research was supported by funds from NSF Grant DCR-84-11954 (Karp), ARPA N00014-80-C-0622 (Leighton), AFOSR 82-0326 (Leighton), NSF Presidential Young Investigator Award with matching funds from Xerox and IBM (Leighton), NSF Grant MCS-80-06938 (Rivest), Semiconductor Research Corporation Grant 82-11-008 (Thompson), NSF Grant MCS 82-04506 (U. V. Vazirani), NSF Grant BCR 85-03611 (U. V. Vazirani), and an IBM Graduate Fellowship (V. V. Vazirani).

[2] Computer Science Division, 573 Evans Hall, University of California, Berkeley, CA 94720, USA.

[3] Mathematics Department and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.

[4] Computer Science Department, Upson Hall, Cornell University, Ithaca, NY 14853, USA.

Received March 18, 1986; revised July 10, 1986. Communicated by C. K. Wong.

Row \ Column	1	2	3	4
1	6	6	7	8
2	5	2	1	3
3	7	4	1	2
4	4	8	5	3

Fig. 1. A global routing problem.

2. Compute a detailed routing that specifies for each net the exact position of each wire, which follows the previously computed global routing and satisfies the usual separation constraints between wires, etc.

In this paper we are concerned exclusively with the problem of finding good global routings (which we henceforth call routings).

1.1. T-turn Routings. We are particularly concerned with t-turn routings, in whch the path for each net contains at most t turns. A one-turn routing will have for each wire either a straight wire segment or an "L"-shaped wire segment. In general, a t-turn routing consists of at most $t+1$ straight-line segments. When horizontal and vertical wires are implemented on distinct layers, then at most t "vias" or "contact cuts" are required to join the straight-line wire segments together.

NOTATION. We denote the set of global routings for problem instance P by $\Gamma(P)$. The set of t-turn global routings is denoted by $\Gamma_t(P)$.

1.2. Example. Figure 1 presents an example of our global routing problem on a 4×4 grid with eight nets. Figure 2 presents a typical solution to this problem, which happens to be in $\Gamma_1(P)$.

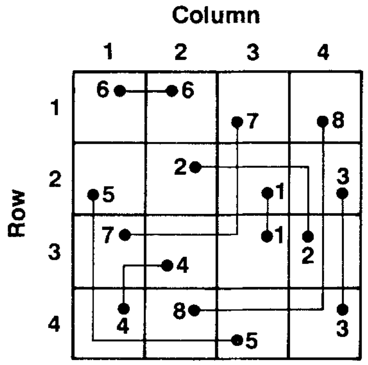

Fig. 2. A solution to the problem of Figure 1.

1.3. Channel Widths. Let P denote an instance of our global routing problem and let R denote a global routing solving P.

NOTATION. Let $w(R)$ denote the maximum number of wires passing from one cell into an adjacent one in the global routing R.

REMARKS. Intuitively, $w(R)$ is the "channel width" which is needed to route the wires of the solution R, so we call w the "width" of the solution R. The one-turn routing R of Figure 2 has width 3, as there are three wires between cell $(2, 4)$ and cell $(3, 4)$. Flipping either net 2 or net 8 to its other "L" configuration will reduce the width to 2. Readers can convince themselves that no one-turn routing has width 1 by considering nets 1, 6, and 7.

DEFINITION. An optimum global routing R is one that minimizes $w(R)$ over all global routings for the given problem instance (i.e., over all $R \in \Gamma(P)$).

NOTATION. We let

$w(P)$ denote the width of an optimal routing for P,
$w_t(P)$ denote the least width of any t-turn routing R that solves P,
$w(n)$ denote the maximum width of any problem instance defined on an $n \times n$ array,
$w_t(n)$ denote the maximum of $w_t(P)$ for any problem instance P defined on an $n \times n$ array,
$w(n, p)$ denote $w(n)$ when p, the number of pins per cell, is not equal to one, and
$w_t(n, p)$ denote $w_t(n)$ when $p \neq 1$.

REMARKS. The reader will be able to distinguish the notations $w(R)$, $w(P)$, and $w(n)$ by the type of argument.

1.4. Motivation. Our research was motivated by the following intriguing conjecture.

CONJECTURE. $w(n) = w_1(n) = \lfloor n/2 \rfloor + 1$.

This controversial-sounding conjecture states that in the worst case we need only consider one-turn routings.

On the other hand, the conjecture merely requires that for any problem instance P there exists a one-turn routing R for P such that $w(R) \leq w(n)$ and not that $w(R) \leq w(P)$. (It is not difficult to develop problem instances P for which $w(P) = 1$ but $w_1(P) = \Omega(n)$.)

2. Results.

Our major theorems are listed here; except for Theorem 1, all proofs and proof sketches are given later.

THEOREM 1. $\lfloor n/2 \rfloor \leq w(n) \leq n$.

PROOF. For the lower bound connect (i, j) to $(i, n-j)$ for $1 \leq j \leq n/2$, and consider the number of wires that must cross from column $\lfloor n/2 \rfloor$ to $\lfloor n/2 \rfloor + 1$. For the upper bound use any routing in $\Gamma_1(P)$ for a given instance P. □

THEOREM 2. $\lfloor n/2 \rfloor + 1 \le w_1(n) \le \lceil n/2 \rceil + 1$. *Furthermore, a one-turn routing* R *with* $w(R) \le \lceil n/2 \rceil + 1$ *can be computed in time* $O(n^3 \log n)$.

REMARKS. Theorem 2 proves the second equality of our conjecture for all even *n*. The upper-bound proof involves the development of an elegant algorithm of independent interest for computing a good integral approximation to the solution of a set of linear equalities. The following theorem states the main result used.

THEOREM 3 (The Rounding Theorem). *Let* $\mathbf{A}$ *be a real-valued* $r \times s$ *matrix, let* $\mathbf{x}$ *be a real-valued s-vector, let* $\mathbf{b}$ *be a real-valued r-vector such that* $\mathbf{Ax} = \mathbf{b}$, *and let* Δ *be a positive real number such that in every column of* $\mathbf{A}$,

(i) *the sum of the positive elements is* $\le \Delta$, *and*
(ii) *the sum of the negative elements is* $\ge -\Delta$.

Then we can compute an integral s-vector $\hat{\mathbf{x}}$ *such that*

(i) *for all i,* $1 \le i \le s$, *either* $\hat{\mathbf{x}}_i = \lfloor \mathbf{x}_i \rfloor$ *or* $\hat{\mathbf{x}}_i = \lceil \mathbf{x}_i \rceil$ (*i.e.,* $\hat{\mathbf{x}}$ *is a "rounded" version of* $\mathbf{x}$), *and*
(ii) $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, *where* $\hat{\mathbf{b}}_i - \mathbf{b}_i < \Delta$ *for* $1 \le i \le r$. *In the case that all entries in* $\mathbf{A}$ *are integers, then a stronger bound applies*: $\hat{\mathbf{b}}_i - \lceil \mathbf{b}_i \rceil \le \Delta - 1$.

Furthermore, if $\mathbf{x}$ contains d distinct components, we obtain the integral approximation $\hat{\mathbf{x}}$ in time $O(r^3 \log(1 + s/r) + r^3 + d^2 r + sr)$.

REMARKS. The Rounding Theorem says that when $\mathbf{A}$ has only a few small nonzero entries in each column, then we can effectively round $\mathbf{x}$ to a nearby *integral* point $\hat{\mathbf{x}}$ while keeping $\mathbf{A}\hat{\mathbf{x}}$ from increasing very much over $\mathbf{Ax}$.

In our application, the initial vector $\mathbf{x}$ is trivially formed as $(1/2, 1/2, \ldots, 1/2)^T$, so that its integral approximation is obtained rapidly. In cases where the initial $\mathbf{x}$ has no repeated entries, our technique will compute an integral $\mathbf{x}$ more slowly.

Some years ago, Beck and Fiala [BF] obtained a theorem with a similar appearance. To the best of our knowledge, our Rounding Theorem cannot be obtained by their proof technique, nor does their result follow from our proof. For the readers' convenience, we quote Beck and Fiala's theorem below.

THEOREM [BF]. *Let* $\mathbf{A}$ *be a* 0–1 $r \times s$ *matrix, let* $\mathbf{x}$ *be a real-valued s-vector, and let* $\mathbf{b}$ *be a real-valued r-vector such that* $\mathbf{Ax} = \mathbf{b}$. *Furthermore, let* Δ *be a positive real number such that in every column of* $\mathbf{A}$, *the sum of the elements is* $\le \Delta$. *Then one can find a* 0–1 *s-vector* $\hat{\mathbf{x}}$ *such that*

(i) *for all i,* $1 \le i \le s$, *either* $\hat{\mathbf{x}}_i = \lfloor \mathbf{x}_i \rfloor$ *or* $\hat{\mathbf{x}}_i = \lceil \mathbf{x}_i \rceil$ (*i.e.* $\hat{\mathbf{x}}$ *is a "rounded" version of* $\mathbf{x}$), *and*
(ii) $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, *where* $|\hat{\mathbf{b}}_i - \mathbf{b}_i| \le \Delta - 1$ *for* $1 \le i \le r$.

REMARKS. Note that Beck and Fiala's matrix $\mathbf{A}$ contains no negative entries. This permits a two-sided error bound on $\mathbf{A}\hat{\mathbf{x}}$, as opposed to our one-sided bound.

THEOREM 4. *It is NP-complete to determine, given an instance P of our global routing problem, whether* $w_1(P) \leq \lceil n/2 \rceil - 2$.

REMARKS. This result is perhaps surprising in view of Theorem 2; the approximation algorithm presented there is remarkably good. The result can also be improved, although we do not include the details here. In particular, it is also NP-complete to determine whether $w_1(P) \leq \lceil n/2 \rceil - 1$. When p is even, it is NP-complete to determine whether $w_1(P) < pn/2$. Given the result proved in Theorem 5, this result is as tight as possible.

THEOREM 5. *When p is even,* $w_1(n, p) = pn/2$.

COROLLARY. $\lfloor n/2 \rfloor \leq w_3(n) \leq \lceil n/2 \rceil + 1$.

COROLLARY. *When p is odd,* $p \lfloor n/2 \rfloor \leq w_3(n, p) \leq \lceil pn/2 \rceil + p$.

REMARKS. Theorem 5 shows that three-turn routings can yield an improvement (by one). The upper bound proof uses an elegant argument based on finding Eulerian tours in an associated graph.

THEOREM 6. *There is a polynomial time approximation algorithm achieving* $w(R) = O(w(P) \log(np/w(P)))$ *for any problem instance P.*

REMARKS. The proof of Theorem 6 involves a hierarchical bottom-up approach, using a recursion based on 2×2 subdivisions. We believe it is possible to reduce the logarithmic term to a constant, but have not yet been able to do so. The result is also valid for P containing multipoint nets.

THEOREM 7. *If* $n \equiv 2$ *or* $3 \pmod 4$, *then* $w(n) \geq \lfloor n/2 \rfloor + 1$.

REMARKS. This lower bound extends that of Theorem 2 to handle routings containing arbitrarily many turns, in the cases indicated.

THEOREM 8. $w_2(n) \leq \lfloor n/2 \rfloor + 1$.

REMARKS. This theorem refines the techniques and results of Theorem 5 and its first corollary, moving from three-turn to two-turn nets and improving the upper bound for odd n by one.

3. Discussion of the Model.

Chen *et al.* [CFKNS] give an excellent overview of how IBM uses algorithms for solving this global routing problem to automatically wire-master-slice logic arrays for their System/370 implementations. The model is particularly appropriate for gate-array technologies where each cell might contain a single NAND gate. Fabrication turn-around time can be very small here since wafers can be preprocessed to contain the array of gates and the only processing required once the logic design is finished is to produce horizontal and vertical wiring on the last two metal layers, to connect the gates together as desired. However, the preprocessing involved usually fixes an upper bound on the value of $w(R)$ that will be allowed—if all routing channels between gates have width 20 then the routing cannot be realized if $w(R) > 20$.

As noted earlier, our concern is with "worst-case" values $w(n)$; in practice one would expect "typical" chips to have $w(P)$ substantially less than $w(n)$.

4. Related Work.

Much of the earlier algorithmic work on global routing (e.g., [HN]), as well as most industrial practice, is based on shortest-path algorithms that route one net at a time. Global routing problems have also been solved with simulated annealing [VK], multicommodity flow [NSS], and hierarchical approaches [BP], [LM].

Johnson [J] gives an overview of the NP-completeness results known in this area. Most notably, Kramer and van Leeuwen prove that (local) wire-routing in gate-arrays is NP-complete if knock-knees are not allowed [KvL].

Some probabilitic models have been developed by El Gamal [EG] to estimate $w(P)$ under various assumptions about the average distance between the pins on a net, etc., in a typical instance P.

Subsequent to our first publication of this research [KLRTVV], two groups have used integer programs to wire actual gate-arrays. (As noted previously, the approach reported here is limited to worst-case analysis.) Hu and Shing [HS] solve the integer program for global routing in a top-down hierarchical fashion. Their approach is thus faster, but lacks the performance guarantees, of the "randomized-rounding" integer programming algorithm currently under investigation by Ng, Raghavan, and Thompson [RT], [NRT].

5. Proofs of Theorems 2–8.

Theorem 1 was proved in Section 2. All other theorems are proved below.

5.1. Proof of Theorem 2: One-Turn Routing by Rounding

THEOREM 2. $\lfloor n/2 \rfloor + 1 \le w_1(n) \le \lceil n/2 \rceil + 1$. *Furthermore, a one-turn routing R with $w(R) \le \lceil n/2 \rceil + 1$ can be computed in time $O(n^3 \log n)$.*

PROOF. For $n = 2$, a lower-bound example is easily constructed as in Figure 3. For larger values of n, we connect the corners of the array as in the 2×2 example. We then put $(\lfloor n/2 \rfloor - 1)$ nets into the outermost horizontal and vertical channels, as shown in Figure 4.

The upper bound is proved using the Rounding Theorem. We first describe how to apply the Rounding Theorem to our routing problem, and then in the next section describe a surprisingly efficient "rounding algorithm."

We assume for convenience here that n is even. Let x_i be a 0–1 valued variable associated with net i indicating which of the two L-shaped routes will be used.

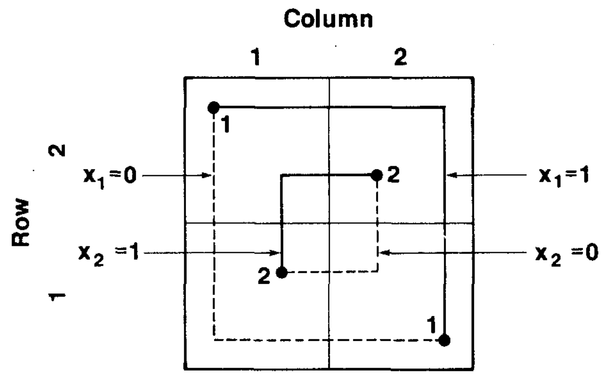

Fig. 3. Lower-bound example for $n = 2$.

The interpretation is fixed but arbitrary. We assume here that each L-shaped route has *exactly* two wire segments. If both pins for a net lie in the same row or column we assume the two L-shaped routes are distinguished by the inclusion of different zero-length wire segments at their ends. (These are degenerate L-shapes with one leg of the L having zero length.) Each assignment of 0–1 values to $\mathbf{x} = (x_1, \ldots, x_{n^2/2})$ places an easily computed number of wire segments in each row and column. For example, in the problem of Figure 3 the number of wire segments in column 1 is $(1 - x_1) + x_2$.

It is then simple to write a set of equations specifying that each row and column will contain exactly $n/2$ wire segments:

$$(*) \qquad \mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A}$ is a $(2n) \times (n^2/2)$ integer-valued matrix, and $\mathbf{b} = (n/2, n/2, \ldots, n/2)^T$. Each variable x_i will participate in at most four constraints, since its two L-routes affect the wire segment count in at most two rows and two columns. Furthermore, it is easy to check that $\mathbf{A}$ satisfies the conditions of the Rounding Theorem with $\Delta = 2$, since each x_i will enter two constraints positively and two negatively.

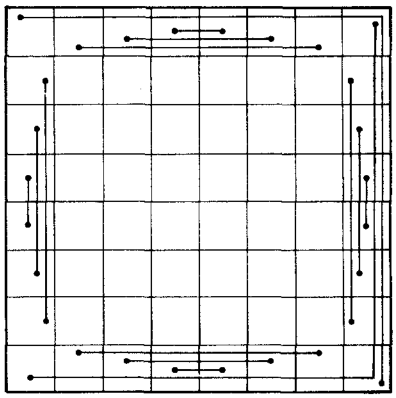

Fig. 4. Lower-bound example for $n = 8$.

Finally, it is easy to see that the vector $\mathbf{x}=(1/2, 1/2, 1/2, \ldots, 1/2)^T$ satisfies equation $(*)$, since each net endpoint will then add 1/2 to the wire segment count for its row and column.

Following the constructive proof of the Rounding Theorem (see Section 5.2), we find a 0–1 valued vector $\hat{\mathbf{x}}$ such that for all i,

$$(\mathbf{A}\hat{\mathbf{x}})_i \leq [\mathbf{b}_i] + \Delta - 1.$$

Since $\Delta = 2$, the vector $\hat{\mathbf{x}}$ corresponds to a routing with at most $\lceil n/2 \rceil + 1$ wires in each row or column. We conclude that $w_1(n) \leq \lceil n/2 \rceil + 1$.

In our application of the Rounding Theorem, we have $r = 2n$ (one equality for each row or column) and $s = n^2/2$ (one variable for each net), so the execution time is $O(n^3 \log n)$. This compares favorably with the more usual approach based on shortest paths, which routes an $n \times n$ array in time $O(n^4)$. □

5.2. *Proof of Theorem 3: The Rounding Algorithm*

THEOREM 3 (The Rounding Theorem). *Let* $\mathbf{A}$ *be a real-valued* $r \times s$ *matrix, let* $\mathbf{x}$ *be a real-valued s-vector, let* $\mathbf{b}$ *be a real-valued r-vector such that* $\mathbf{Ax} = \mathbf{b}$, *and let* Δ *be a positive real number such that in every column of* $\mathbf{A}$,

(i) *the sum of the positive elements is* $\leq \Delta$, *and*
(ii) *the sum of the negative elements is* $\geq -\Delta$.

Then we can compute an integral s-vector $\hat{\mathbf{x}}$ *such that*

(i) *for all i,* $1 \leq i \leq s$, *either* $\hat{\mathbf{x}}_i = \lfloor \mathbf{x}_i \rfloor$ *or* $\hat{\mathbf{x}}_i = \lceil \mathbf{x}_i \rceil$ (*i.e.* $\hat{\mathbf{x}}$ *is a "rounded" version of* $\mathbf{x}$), *and*
(ii) $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, *where* $\hat{\mathbf{b}}_i - \mathbf{b}_i < \Delta$ *for* $1 \leq i \leq r$. *In the case that all entries in* $\mathbf{A}$ *and* $\mathbf{b}$ *are integers, then a stronger bound applies*: $\hat{\mathbf{b}}_i - \lceil \mathbf{b}_i \rceil \leq \Delta - 1$.

Furthermore, if $\mathbf{x}$ contains d distinct components, we obtain the integral approximation $\hat{\mathbf{x}}$ in time $O(r^3 \log(1 + s/r) + r^3 + d^2 r + sr)$.

PROOF. We exhibit a "rounding algorithm" that efficiently computes an appropriate vector $\hat{\mathbf{x}}$, given $\mathbf{A}$, Δ, $\mathbf{b}$, and $\mathbf{x}$ as input.

In brief, our approach is to convert the given problem to a 0–1 problem, by subtracting off the integer parts of the given vector $\mathbf{x}$. We then examine the $r \times s$ matrix $\mathbf{A}$. If $r < s$, clearly $\mathbf{A}$ is singular. We are thus able to round some x_i without affecting the product $\mathbf{Ax}$. Similarly, if $s < r$, we can eliminate some constraint b_j. The only difficult case is when $s = r$. By a combinatorial argument, we show how to find a constraint that can be safely ignored, in the sense that no rounding of the $\mathbf{x}$ can cause this constraint to be violated by more than Δ.

The Rounding Algorithm:

1. [*Convert to 0–1 problem.*] Replace $\mathbf{x}$ by $\mathbf{x} - \mathbf{x}'$, where $\mathbf{x}'_i = \lfloor x_i \rfloor$ for all i. Replace $\mathbf{b}$ by $\mathbf{b} - \mathbf{Ax}'$. Solve the modified problem (steps 2 to 3) and then convert back by adding $\mathbf{x}'$ to the $\hat{\mathbf{x}}$ computed and $\mathbf{Ax}'$ to the $\mathbf{A}\hat{\mathbf{x}}$ computed. Halt.

$$C = \left(\begin{array}{cccccc|c} 1 &&&&&& \\ & 1 && 0 &&& \\ && 1 &&&& \alpha \\ &&& 1 &&& \\ & 0 &&& 1 && \\ &&&&& 1 & \end{array}\right)$$

Fig. 5. Row-echelon form of matrix **C**.

2. [*Fast reduction in the number of variables.*] If there are d distinct values among the variables $\mathbf{x}$, this step reduces the number of variables to r or less in $O(d + r\log(1 + s/r))$ iterations.
   2a. [*Test if done.*] If $s \le r$ go to step 3.
   2b. [*Grouping.*] Divide the s variables into $k = \max(d, r+1)$ groups, where the variables in each group all have the same value. (If $d < r+1$, then make the k groups as nearly equal in size as possible.) Consider a new problem $\mathbf{Cy} = \mathbf{b}'$, where $\mathbf{y}$ is a k-vector having one element for each group, and $\mathbf{C}$ is an $r \times k$ matrix. Here, $\mathbf{C}$, $\mathbf{y}$, and $\mathbf{b}'$ are obtained from $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{b}$ by adding the constraint that within each group each variable will have the same value. For example, the first column of $\mathbf{C}$ is the sum of the columns of $\mathbf{A}$ corresponding to variables in the first group, and $\mathbf{y}_1$ is the (common) value of the $\mathbf{x}_i$'s from the first group.
   2c. [*Reduce* **C** *to row-echelon form.*] Using elementary row operations, convert the $r \times k$ matrix $\mathbf{C}$ to row-echelon form, as in Figure 5 (drawn for the case that $\mathbf{C}$ has rank $r = 6$). Note that this operation does not change the null space of $\mathbf{C}$.
   2d. [*Round.*] Let $\mathbf{y}_0$ be a nonzero k-vector in the null space of $\mathbf{C}$. (This is easy to compute given step 2c.) Let $\lambda^* = \min\{\lambda \ge 0 \mid \mathbf{y} + \lambda\mathbf{y}_0$ has an integral component$\}$ and let $\mathbf{w} = \mathbf{y} + \lambda^*\mathbf{y}_0$.
   2e. [*Update.*] For each variable $\mathbf{x}_i$ in a group j where $\mathbf{w}_j$ is integral, fix $\hat{\mathbf{x}}_i$ at $\mathbf{w}_j$ and remove $\mathbf{x}_i$ from the problem (i.e., set $\mathbf{b} = \mathbf{b} - \mathbf{w}_j\mathbf{A}_{[*,i]}$ where $\mathbf{A}_{[*,i]}$ is the ith column of $\mathbf{A}$, delete $\mathbf{x}_i$ from $\mathbf{x}$ and delete the ith column of $\mathbf{A}$). Set the remaining $\mathbf{x}_i$ to their group values $\mathbf{w}_j$.
   2f. [*Revise group structure.*] If now $s \le r$, go to step 3. If there are fewer than $(r+1)$ groups, repeatedly split the largest group into two smaller ones until there are exactly $(r+1)$ groups. Update $\mathbf{C}$ to reflect the changes in steps 2e and 2f, and return to step 2d.
3. [*One by one reduction in equalities and variables.*] If $s < r$ execute step 3a, if $s > r$ execute step 3b, else execute step 3c. Repeat step 3 until all variables have been fixed. Then halt; the desired solution has been found.
   3a. [$s < r$: *Eliminate an equality.*] Let $\mathbf{I}$ be the set of 2^s integer vectors obtained by rounding each component $\mathbf{x}_j$ either up to $\lceil \mathbf{x}_j \rceil$ or down to $\lfloor \mathbf{x}_j \rfloor$. As shown below, an argument involving an interchange of two summations shows that for some i, $(\mathbf{Az})_i < (\mathbf{Ax})_i + \Delta$ for all $\mathbf{z} \in \mathbf{I}$. This means we can drop the ith inequality from explicit consideration, since $\hat{\mathbf{b}}_i - \mathbf{b}_i$ will be less than Δ no matter how the rounding is done. Also, in the case that all

entries of $\mathbf{A}$ and $\mathbf{b}$ are integral, then all entries in $\hat{\mathbf{b}}$ must be integral. Thus we have the stronger result that $\hat{\mathbf{b}}_i - \lceil \mathbf{b}_i \rceil \le \Delta - 1$. The existence of a redundant inequality i is shown as follows:

$$\begin{aligned}
\sum_i \max_{\{\mathbf{Z}\in\mathbf{I}\}} ((\mathbf{Az})_i - (\mathbf{Ax})_i) &= \sum_i \left(\sum_{\{j|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij}(1-\mathbf{x}_j) + \sum_{\{j|\mathbf{a}_{ij}<0\}} -\mathbf{a}_{ij}(\mathbf{x}_j) \right) \\
&= \sum_j \left(\sum_{\{i|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij}(1-\mathbf{x}_j) + \sum_{\{i|\mathbf{a}_{ij}<0\}} -\mathbf{a}_{ij}\mathbf{x}_j \right) \\
&\le \sum_j (\Delta(1-\mathbf{x}_j) + \Delta\mathbf{x}_j) \\
&\le \sum_{1\le j\le s} \Delta \\
&< r\Delta.
\end{aligned}$$

Hence there exists an i, $1 \le i \le r$, such that $\forall \mathbf{z}\ (\mathbf{Az})_i < (\mathbf{Ax})_i + \Delta$. We can find a redundant row i rapidly, since

$$\begin{aligned}
\max_{\{\mathbf{Z}\in\mathbf{I}\}} ((\mathbf{Az})_i - (\mathbf{Ax})_i) &= \sum_{\{j|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij}(1-\mathbf{x}_j) + \sum_{\{j|\mathbf{a}_{ij}<0\}} -\mathbf{a}_{ij}\mathbf{x}_j \\
&= -\sum_j \mathbf{a}_{ij}\mathbf{x}_j + \sum_{\{j|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij} \\
&= -\mathbf{b}_i + \sum_{\{j|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij}.
\end{aligned}$$

Thus we merely look for an i such that

$$-\mathbf{b}_i + \sum_{\{j|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij} < \Delta.$$

3b. [$s > r$: *Eliminate a variable.*] This is much as in steps 2d–2f, except we may only eliminate one variable; here each variable is in its own group.

3c. [$s = r$: *Eliminate an equality or a variable.*] As will be proved below, there are two possibilities. Either the rows of $\mathbf{A}$ are linearly dependent, or $\exists i\ \forall \mathbf{z}\ (\mathbf{Az})_i < (\mathbf{Ax})_i + \Delta$. Thus we can eliminate either a variable as in step 3b or an equality as in step 3a. We argue as follows. If $\forall i\ \forall \mathbf{z}\ (\mathbf{Az})_i \ge (\mathbf{Ax})_i + \Delta$, so that we cannot eliminate an equality, then

$$\begin{aligned}
r\Delta &\le \sum_i \max_{\{\mathbf{Z}\in\mathbf{I}\}} ((\mathbf{Az})_i - (\mathbf{Ax})_i) \\
&= \sum_j \left(\left(\sum_{\{i|\mathbf{a}_{ij}>0\}} \mathbf{a}_{ij} \right) (1-\mathbf{x}_j) + \left(\sum_{\{i|\mathbf{a}_{ij}<0\}} -\mathbf{a}_{ij} \right) \mathbf{x}_j \right) \\
&\le \sum_j (\Delta(1-\mathbf{x}_j) + \Delta(\mathbf{x}_j)) \\
&= r\Delta.
\end{aligned}$$

Since $\forall j\ 0 < \mathbf{x}_j < 1$, the second "$\leq$" in the derivation above is an equality only if

$$\forall j \quad \sum_{\{i \mid \mathbf{a}_{ij} > 0\}} \mathbf{a}_{ij} = \Delta$$

and

$$\forall j \quad \sum_{\{i \mid \mathbf{a}_{ij} < 0\}} -\mathbf{a}_{ij} = \Delta.$$

Thus

$$\forall j \quad \sum_i \mathbf{a}_{ij} = 0,$$

demonstrating that the rows of **A** are linearly dependent.

This completes our description of the rounding algorithm. The claimed running time is verified as follows.

Each iteration of step 2 eliminates one group of variables. We distinguish between the original d groups and the groups that are obtained by splitting other groups (in step 2f or 2a). Because we always split the largest remaining group, the smallest nonoriginal group contains at least $\lfloor l/2 \rfloor$ variables, where l is the size of the largest group before the split. After d iterations, there are at most $r+1$ groups and the smallest nonoriginal group contains at least $1/(3r)$ of the remaining variables. We have at most r variables (and thus exit step 2) after j iterations, where j is constrained by the following equation for the case $j \geq d$:

$$s\left(1 - \frac{1}{3r}\right)^{j-d} \leq r.$$

Taking logarithms of both sides, and using the fact that $-\ln(1 - 1/z) > 1/z$ for $z > 1$, we obtain

$$j \leq \max\{d, d + 3r \ln(s/r)\} \leq d + 3r \ln(1 + s/r)).$$

Thus there are $O(d + \log(1 + s/r))$ iterations of step 2. Also, there are at most $2r$ iterations of step 3.

We now show that each iteration on a $k \times r$ matrix (where $k \geq r$) can be carried out in time $O(kr)$. Since **C** is in row-echelon form, we can read off a nonzero element of its null space in $O(kr)$ time. This time is also sufficient to maintain **C** in row-echelon form.

Total time for the first d iterations is then $O(d^2 r)$, and the total time for the last $O(r \log(1 + s/r))$ iterations is $O(r^3 \log(1 + s/r))$. The stated total time for the algorithm also allows $O(d^2 r + sr)$ time to transform the original matrix **C** into row-echelon form, in step 2c, and $O(r^3)$ time for step 3. □

5.3. Proof of Theorem 4: NP-Completeness of Optimal One-Turn Routing

THEOREM 4. *It is* NP-*complete to determine, given an instance P of our global routing problem, whether* $w_1(P) \leq \lceil n/2 \rceil - 2$.

Proof. The reduction is from 3-SAT. Given an instance E of 3-SAT with variables $x_1, x_2, \ldots, x_q$ and clauses $c_1, c_2, \ldots, c_m$, set $n = 14m+3$ and define the routing problem P as follows.

Pins in the rightmost $7m+3$ columns of the grid are not included in any net. Any one-turn routing of P can thus have row widths at most $7m = \lceil n/2 \rceil - 2$. Therefore, we are only concerned with column widths in what follows.

Pins in the leftmost $7m$ columns but not in the middle seven rows are paired so that the pin in the ith row of the jth column is linked to the pin in the $(n-i+1)$st row of the jth column. Each of these nets must be routed as a vertical wire, and the question of whether $w_1(P) \le \lceil n/2 \rceil - 2$ is equivalent to the question of whether the middle seven rows can be routed with column width 2.

The middle seven rows of the leftmost m columns are used to represent the clauses. There is one column for each clause. The middle seven rows of the next $6m$ columns are used to represent the variables. There are $2r_i$ columns for variable x_i, where r_i is the number of times x_i appears in E.

The structure of the "clause columns" is trivial. A unique pin name c_{jk} is assigned to each of the three terms in clause c_j. See Figure 6.

The "variable columns" are more complex. As indicated in Figure 6, we stitch a net d_{il} $(1 \le l \le 2r_i)$ vertically through each of the $2r_i$ columns for variable x_i. We stitch nets a_{il} and b_{il} $(1 \le l \le r_i)$ in a knight's-move pattern through adjacent pairs of columns. Finally, we connect variables to clauses in a fashion that depends on the given instance E of 3-SAT. If the kth term in clause c_j $(k = 1, 2,$ or $3)$ is x_i, then the first available $+$ symbol in the $2r_i$ columns for x_i is replaced by c_{jk}. If the kth term in clause c_j is $\overline{x_i}$, then the first available $-$ symbol in the $2r_i$ columns for x_i is replaced by c_{jk}. Since x_i appears r_i times in E, there are always enough $+$'s and $-$'s for all the c_{jk}'s. The remaining $+$'s and $-$'s (as well as the dots) are not assigned to a net. In what follows, we show that the middle seven rows can be routed with column width 2 if and only if E is satisfiable.

Since we allow at most one bend in a routing, the nets labeled with d_{ik}'s must

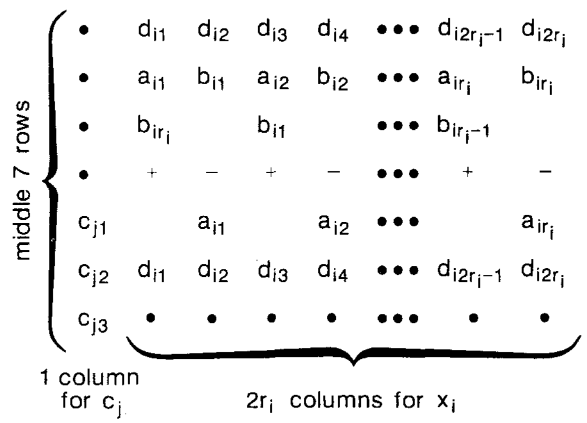
middle 7 rows
• d_i1 d_i2 d_i3 d_i4 ••• d_i2r_i−1 d_i2r_i
• a_i1 b_i1 a_i2 b_i2 ••• a_ir_i b_ir_i
• b_ir_i b_i1 ••• b_ir_i−1
• + − + − ••• + −
c_j1 a_i1 a_i2 ••• a_ir_i
c_j2 d_i1 d_i2 d_i3 d_i4 ••• d_i2r_i−1 d_i2r_i
c_j3 • • • • ••• • •
1 column for c_j
2r_i columns for x_i

Fig. 6. An NP-complete routing problem P.

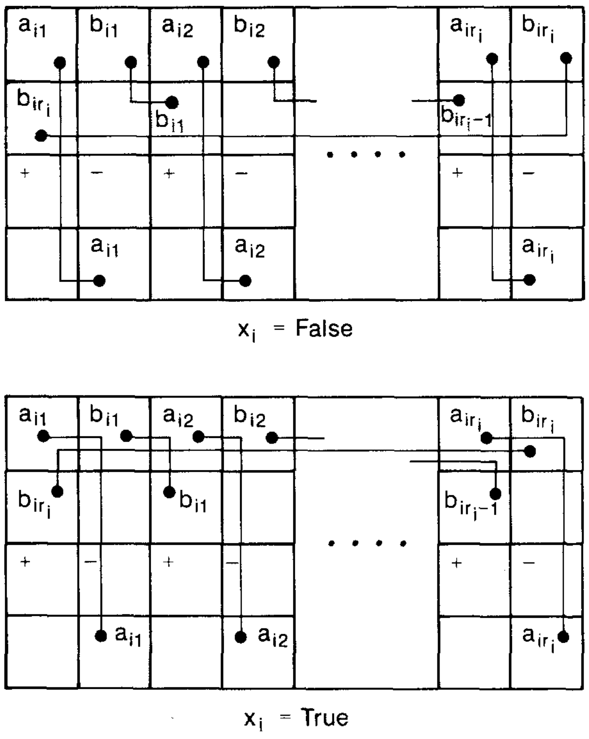

Fig. 7. Routing the NP-complete problem P.

be routed as vertical wires. This leaves only two ways to route the nets labeled with a_{ik}'s and b_{ik}'s. The two routings correspond in a natural way to the truth value of the associated variable x_i. The routings are shown in Figure 7.

It remains to route the c_{jk}'s. It is easily shown that if c_{jk} corresponds to x_i where x_i has a true routing or to $\overline{x_i}$ where x_i has a false routing, then net c_{jk} can be safely routed without using a vertical wire segment in the column for c_j. This is not the case if c_{jk} corresponds to x_i where x_i has a false routing or to $\overline{x_i}$ where x_i has a true routing. In the latter cases, the net for c_{jk} must include a vertical wire segment in the column for c_j that passes through the top of the cell containing c_{j1}. Hence the middle seven rows can be routed with column width 2 if the only if there is a k for each j such that c_{jk} corresponds to x_i where x_i has a true routing or to $\overline{x_i}$ where x_i has a false routing. This condition is equivalent to E being satisfiable. □

5.4. Proof of Theorem 5: Routing Using Eulerian Tours

THEOREM 5. *When p is even, $w_1(n, p) = pn/2$.*

PROOF. Let each of the cells of the $n \times n$ array be the vertices of a graph, and connect any two vertices that are connected by a net.

Since p is even, every vertex will have an even degree.

Thus the edges can be organized into a direct path which is an Eulerian tour, traversing each edge exactly once. (The case that the graph is not connected arises but is easy to handle . . .)

For each edge $(i, j) \to (k, l)$ of the Eulerian tour, we use an L-shaped route with the horizontal arc first:

$$(i, j) \to (i, l) \to (k, l).$$

Since each vertex will have $p/2$ horizontal arcs leaving it and $p/2$ vertical arcs entering it, we can route the entire chip with $pn/2$ tracks in each row or column.

To prove the first corollary ($p = 1$), we group the cells into 2×2 squares and apply the above construction for $p = 4$.

Then we may need to introduce small (length 1) jogs within each square to get the two horizontal arcs leaving on different rows. This we can do with only one extra track for each row or column, yielding $\lceil n/2 \rceil + 1$ tracks at most. Here each L-shaped route may have a little tail at each end so a net may have three turns total.

In general, for odd p, we use the same idea of grouping cells into 2×2 squares. Assigning routes with Euclidean tours, we put at most pn horizontal arcs on each pair of rows. Using p additional tracks for length-1 jogs within the 2×2 squares, we split the pn arcs evenly between the two rows. In all, we use $\lceil pn/2 \rceil + p$ tracks per row, proving the second corollary. □

5.5. *Proof of Theorem 6: Provably Good Routing*

THEOREM 6. *There is a polynomial time approximation algorithm achieving* $w(R) = O(w(P) \log (np/w(P)))$ *for any problem instance* P.

PROOF. (Sketch): Let $cut(P)$ denote the maximum, over all subsquares of the $n \times n$ array, of the number of nets which must cross the border of the square, divided by the perimeter of that square. It is easy to see that $w(P) \geq cut(P) \geq p$.

Divide the chip into squares with $\lambda = cut(P)/p$ cells on a side. Route these squares independently, in an arbitrary one-turn manner using channel width at most $O(cut(P))$. Nets that must leave a square can be routed arbitrarily to any convenient point (different for each net) on the perimeter of that square. Then proceed through $\log(n/\lambda) \leq \log n$ levels of bottom-up recursion, at each level pasting together four squares from the previous level in a 2×2 pattern.

The statement of the theorem allows $O(cut(P))$ additional width in each channel for new wiring at each level of the recursion. By the definition of $cut(P)$, no more than $cut(P)$ nets will leave any square at any level of the recursion. Thus the recursively generated wiring problems are trivial. Total channel width is $O(cut(P) \log(np/cut(P)))$, which is an increasing function of $cut(P)$. Using $w(P)$ as an upper bound on $cut(P)$, we see that our routing R has channel width $w(R) = O(w(P) \log(np/w(P)))$. A weaker but more understandable bound is $w(R) = O(w(P) \log n)$. □

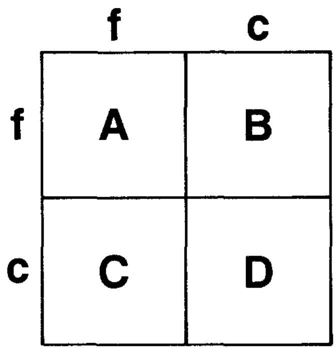

Fig. 8. Four quadrants of a chip.

5.6. Proof of Theorem 7: Improved Lower Bound

THEOREM 7. *If* $n \equiv 2$ *or* 3 (mod 4), *then* $w(n) \geq \lfloor n/2 \rfloor + 1$.

PROOF. Let $f = \lfloor n/2 \rfloor$ and $c = \lceil n/2 \rceil$. Consider dividing the chip as shown in Figure 8 into four quadrants A, B, C, and D, where A is $f \times f$, B and C are $f \times c$, and D is $c \times c$.

Consider a problem instance where each pin of A is to be connected to a corresponding pin in D, and each pin in B is connected with a pin in C. (If $c > f$, the remaining pins in D can be left unattached, or paired off.) Since $|A| = f^2$ is odd, at least $\lceil f^2/2 \rceil$ of the wires from A must run through B (without loss of generality—the case for C is symmetric.) Thus the perimeter of B will be crossed at least $(f^2+1)+fc$ times: (f^2+1) times for the A–D nets and f_C times for the B–C nets. Since the perimeter of B is crossed by only $f+c$ channels (rows of columns), at least one of these channels must contain at least

$$\left\lceil \frac{(f^2+1)+fc}{f+c} \right\rceil = \left\lceil f + \frac{1}{f+c} \right\rceil = f + 1 = \left\lfloor \frac{n}{2} \right\rfloor + 1$$

wires. □

5.7. Proof of Theorem 8. Good Two-Turn Routings

THEOREM 8. $w_2(n) \leq \lfloor n/2 \rfloor + 1$.

PROOF. This is similar to the proof of the first corollary to Theorem 5, except that we group the cells regularly into 1×2 rectangles instead of 2×2 squares. The Eulerian theorem is applied as before. Finally, the L-shaped routings obtained will have to have at most one tail added to produce the final routing. When n is even, it is easy to arrange the tails without increasing the number of tracks required by more than one. When n is odd, the argument is a little more delicate. Consider labeling each pin either "H" or "V" according to whether the route determined by the Eulerian tour would connect to that pin with a horizontal or vertical segment. Figure 9 shows a labeling that might result for a problem with $n = 7$.

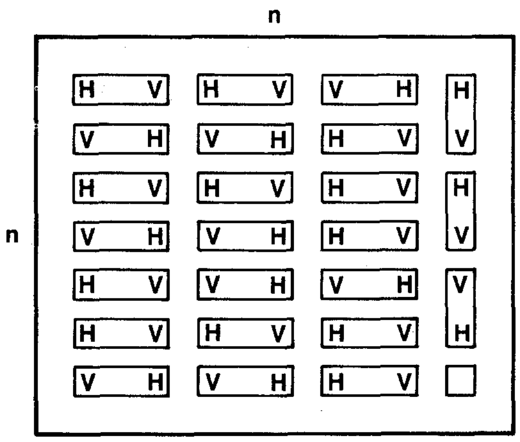

Fig. 9. Labeling of pins for a good two-turn routing.

We are guaranteed that each 1×2 rectangle contains one H and one V by the use of the Eulerian tour, and we need to guarantee that each row has at most $\lfloor n/2 \rfloor + 1$ H's and that each column has at most $\lfloor n/2 \rfloor + 1$ V's. The rows are already okay if the tiling pattern is like that of Figure 9. To adjust the columns we note that by running a short tail within a rectangle we can effectively move a V "on top of" its neighboring H. We can do this safely only in rows which have a V in the rightmost column; otherwise the tail might increase the required channel width. However, there are $\lfloor n/2 \rfloor$ in the rightmost column, so we can always move as many as $\lfloor n/2 \rfloor$ V's out of any column into a neighboring one. Thus we can use the tails to guarantee that no column will have more than $\lfloor n/2 \rfloor + 1$ V's.

6. Open Problems.

We present here some open problems related to the above results.

Open Problem 1. Is there a constant c and a polynomial-time global routing algorithm A such that A will produce for any problem instance P a routing R with $w(R) \leq c \cdot w(P)$ (i.e. a routing whose width is within a constant factor of optimal)?

Open Problem 2. What is $w_t(n) - w(n)$ for any fixed t? Is there a fixed t for which this difference equals 0 for all n?

Open Problem 3. What are other applications of the Rounding Algorithm?

Open Problem 4. Can the logarithmic factor in the running time of the Rounding Algorithm be eliminated?

Open Problem 5. Let $cut(P)$ be defined as in the proof of Theorem 6. Is there a constant c such that $w(P) \leq c \cdot cut(P)$ for all problem instances P? (Note: we can prove that a similar measure computing wire-length within subsquares is linearly related to the *cut* measure.)

Open Problem 6. Can the additive "$+p$" term be improved in the second corollary to Theorem 5?

Open Problem 7. Develop, empirically or otherwise, a good model of the wiring-problem instances that arise in practice.

References

[BF] J. Beck and T. Fiala, "Integer-making" theorems, *Discrete Appl. Math.*, **3** (1981), 1-8.

[BL] S. N. Bhatt and F. T. Leighton, A framework for solving VLSI graph layout problems, *J. Comput. System Sci.*, **28** (1984), 300-343.

[BP] M. Burstein and R. Pelavin, Hierarchical wire routing, *IEEE Trans. Computer-Aided Design*, **2** (1983), 223-234.

[CFKNS] K. A. Chen, M. Feuer, K. H. Khokhani, N. Nan, and S. Schmidt, The chip layout problem: an automatic wiring procedure, *Proceedings of the* 14*th Design Automation Conference*, New Orleans, 1977, pp. 298-302.

[CR] P. Czerwinski and V. Ramachandran, Optimal VLSI graph embeddings in variable aspect ratio rectangles, *Algorithmica* (submitted).

[EG] A. A. El Gamal, Two-dimensional stochastic model for interconnections in master slice integrated circuit," *IEEE Trans. Circuits and Systems*, **28** (1981), 127-138.

[HN] S. J. Hong and R. Nair, Wire routing machines—new tools for VLSI physical design, *Proc. IEEE*, **71** (1983), 57-65.

[HS] T. C. Hu and M. T. Shing, A decomposition algorithm for circuit routing, *Math. Programming Stud.*, **24** (1985), 87-103.

[J] D. S. Johnson, The NP-completeness column: an outgoing guide, *J. Algorithms*, **3** (1982), 381-395.

[KLRTVV] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. Vazirani, and V. Vazirani, Global wire-routing in two-dimensional arrays, *Proceedings of the* 24*th FOCS Conference*, 1983, pp. 453-459.

[KvL] M. R. Kramer and J. van Leeuwen, Wire-routing is NP-complete, Vakgroep Informatica Technical Report RUU-CS-82-4, Rijksuniversiteit Utrecht, 1982, 11 pp.

[L1] F. T. Leighton, New lower-bound techniques for VLSI, *Proceedings 22nd FOCS Conference*, 1981, pp. 1-12.

[L2] F. T. Leighton, A layout strategy for VLSI which is provably good, *Proceedings of the* 14*th ACM STOC Conference*, 1982, pp. 85-98.

[L3] C. E. Leiserson, Area-efficient graph layouts (for VLSI), *Proceedings* 21*st FOCS Conference*, 1980, pp. 270-281.

[LM] J. T. Li and M. Marek-Sadowska, Global routing for gate arrays, *IEEE Trans. Computer-Aided Design*, (1984), 298-308.

[NRT] A. P.-C. Ng, P. Raghavan, and C. D. Thompson, Experimental results for a linear program global router, *Proceedings of the* 23*rd ACM/IEEE Design Automation Conference*, 1986, 659-662.

[NSS] T. Nishizeki, N. Saito, and K. Suzuki, A linear-time routing algorithm for convex grids, *IEEE Trans. Computer-Aided Design*, **4** (1985), 68-76.

[RT] P. Raghavan and C. D. Thompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, *Combinatorica* (to appear).

[T] C. Thompson, "Area-time complexity for VLSI," *Proceedings of the* 11*th ACM STOC Conference*, 1979, pp. 81-88.

[VK] M. P. Vecchi and S. Kirkpatrick, Global wiring by simulated annealing, *IEEE Trans. Computer-Aided Design*, **2** (1983), 215-222.