Minimum Area Joining of Compacted Cells[1]

Seonghun Cho and Sartaj Sahni

Department of Computer and Information Science, University of Florida

Gainesville, FL 32611, USA

Abstract. We consider the problem of joining a row of compacted cells so as to minimize the area occupied by the cells and the interconnects. The cell joining process includes cell stretching and river routing. Since we consider multilayer routing, we first develop necessary and sufficient conditions for feasible multilayer river routing. This is followed by a development of fast algorithms for the multilayer river routing problem. We then propose several heuristics to join a row of cells in such a way that area is minimized. The proposed heuristics are compared, experimentally with that proposed by Cheng and Despain [CHEN89].

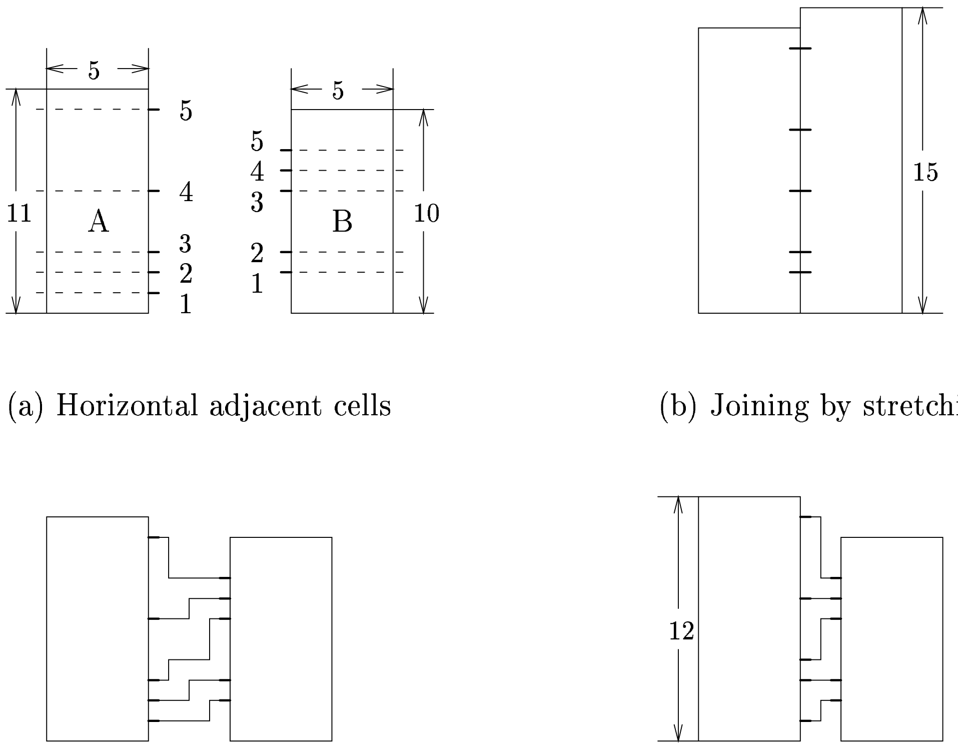
Keywords and Phrases. Cell joining, cell stretching, compacted cells, multilayer river routing, symbolic sticks designs, VLSI design.

1 Introduction

When designing circuits with compacted symbolic sticks basic cells, the circuit is realized by a collection of compacted cells that tile a two-dimensional area. The intercell interconnects are such that each interconnect connects two terminals that are on adjacent boundaries of neighboring cells. So, for example, if cells A and B (Figure 1(a)) are neighboring cells of the circuit, then the right boundary of A is adjacent to the left boundary of B. The number of terminals on each of these boundaries will be the same and the i'th terminal (from the bottom) on the right boundary of A is to be connected to the i'th terminal (from the bottom) on the left boundary of B.

Since the cells are available in compacted form, it is not possible to reduce the distance between any pair of terminals on any side a cell. However, this distance can be increased

[1]This research was supported, in part, by the National Science Foundation under grant MIP-9103379.



(a) Horizontal adjacent cells

(b) Joining by stretching

(c) Joining by river routing

(d) Combination cell joining

Figure 1: Cell Joining

by stretching the cell. In the example of Figure 1(a), we can stretch either cell vertically by defining a horizontal cut line at any position and pulling the two cell pieces apart by any desired amount (the cell can also be stretched horizontally by using a vertical cut line).

The required interconnects between cells A and B of Figure 1(a) can be accomplished by stretching cells A and B so that the terminals of A and B line up as in Figure 1(b). The broken lines in Figure 1(a) indicate the cut lines used for stretching. The stretching enables us to join cells A and B using no routing tracks (by "join" we mean make the interconnects between cells A and B). This method of joining cells is also called pitch matching.

Another way to join cells A and B is to river route the interconnects as in Figure 1(c). This uses routing tracks in a channel between cells A and B but does not increase cell height. The pitch matching and river routing approaches to cell joining have been studied in [BOYE88] and [WEST81]. Algorithms for single-layer river routing can be found in [LEIS83, MIRZ87, PINT82, PINT83].

Cell stretching (or pitch matching) increases the height of the layout while river routing increases its width. Both affect the layout area. The layout of Figure 1(b) has area 150. To compute the area of the layout of Figure 1(c), we assume tracks have unit separation. So, the layout width is 14 and height is 11. The layout has area 154. Cheng and Despain [CHEN89] have proposed using a combination of cell stretching and river routing so as to obtain layouts with smaller area than possible when only one of these joining methods is used. Figure 1(d) shows the result of joining cells A and B using both stretching and river routing. The area of this layout is 144. This is minimum for the instance of Figure 1(a).

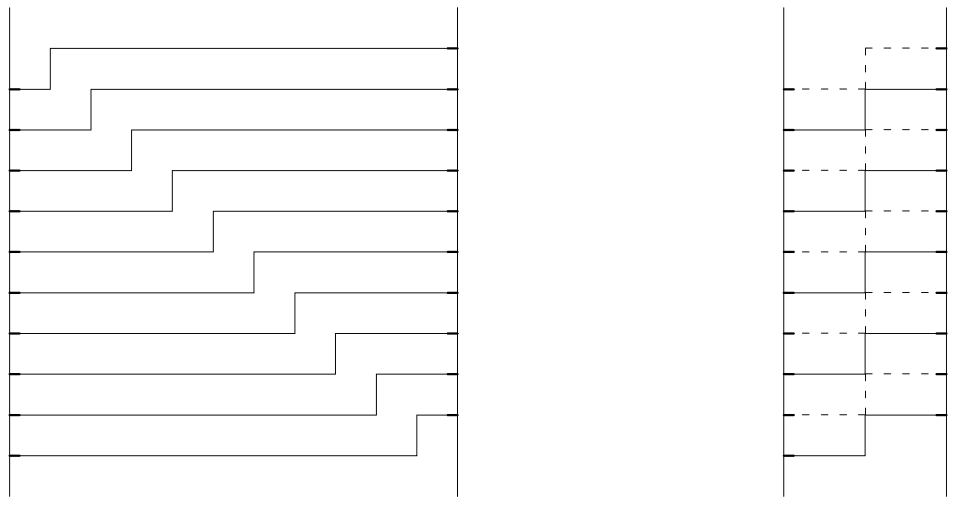
Cheng and Despain [CHEN89] have proposed a heuristic for single layer joining of compacted cells. At each step of their heuristic either a row or column of compacted cells is joined. Following this, the row or column of joined cells is replaced by a composite cell that represents the result of joining. Notice that when a row (column) of cells is joined, cells may be stretched vertically (horizontally) and river routing is done in a vertical (horizontal) channel. To join a row of cells, Cheng and Despain [CHEN89] bound the maximum height to which a cell may be stretched. This bound is

$$h_{max} + h_{avg}^2/(4 * h_{max})$$

where h_{max} is the height of the tallest compacted cell being joined and h_{avg} is the average height of the cells being joined.

Using this bound, cells are joined one-at-a-time using a penalty/reward scheme to determine if a pair of terminals is to be joined by stretching or by river routing.

Lim, Cheng, and Sahni [LIM93] have considered the case when only two cells are to be joined. They develop fast polynomial time algorithms to obtain the minimum area join of two cells. In addition, they are able to obtain, in low order polynomial time, minimum area joins that minimize the length of the longest wire or the total wire length. Lim [LIM92] has proposed an $O(n(n/c)^{c-1})$ algorithm to find the minimum area join of c cells having a total of n terminals. This algorithm does an exhaustive search over all possible numbers of tracks in the $c-1$ routing channels between adjacent cells. A constraint graph is used to determine the minimum height layout for each assignment of number of tracks to routing channels. The time required per track assignment is $O(n)$ and the worst case number of track assignments is $O((n/c)^{c-1})$. The algorithm of [LIM92] is flawed as it handles channels with



(a) 1-layer (b) 2-layer

Figure 2: l-layer river routing

zero routing tracks by joining the adjacent cells using minimum height cell stretching and then considers the joined cells as one. This problem is easily fixed, however, by combining, in the constraint graph, pairs of vertices that represent corresponding terminals of the two cells (i.e., i'th terminals of each cell) with zero routing tracks in between.

In this paper, we consider the case when $l \geq 1$ routing layers are available to river route the inter cell connections. Note that while multiple layers do not affect layout area when cell stretching alone is used, a reduction in area is possible when cell stretching is combined with river routing or when river routing alone is used. We assume that in each layer of each routing channel, the interconnects are to be accomplished using river routing. When the number of layers available for river routing is increased, one may see a dramatic reduction in the number of routing tracks needed per layer. Figure 2 shows an instance that needs n tracks when routed in one layer but only one track/layer when routed in two layers.

We begin, in section 2, by developing a necessary and sufficient condition for a river routing instance to be routable in l layers using at most t tracks per layer. This development includes the formulation of two algorithms to perform l-layer river routing when such a routing is possible. In Section 3, we describe the constraint graph used to determine minimum height stretching of c cells. Heuristics for the minimum area joining of c cells are proposed in

Section 4 and the results of experiments with these are provided in Section 5. Our conclusions appear in Section 6.

2 l-layer River Routing

Let (A_i, B_i), $1 \le i \le m$, be a set of terminal pairs such that the A_i's are on one side (say left or top) of a routing channel and the B_i's are on the other (right or bottom) side. Terminal A_i is to be connected to terminal B_i, $1 \le i \le m$. For this channel routing instance to be an instance of river routing, it must be the case that $a_1 < a_2 < \ldots < a_m$ and $b_1 < b_2 < \ldots < b_m$ where a_i and b_i, respectively, give the positions of terminals A_i and B_i, $1 \le i \le m$. We may assume an underlying grid with each terminal being at a grid position. In the case of a horizontal (vertical) channel the a_is and b_is are grid column (row) numbers. Leiserson and Pinter [LEIS83] have obtained the following necessary and sufficient condition for a river routing instance to be routable in a single layer using at most $t \ge 0$ tracks.

Theorem 1 [LEIS83] *The river routing instance defined above is routable in a single layer using at most $t \ge 0$ tracks if and only if*

(a) $a_{i+t} - b_i \ge t$

(b) $b_{i+t} - a_i \ge t$

for every $i \le m - t$.

For the general case of $l \ge 1$ layers, we obtain the necessary and sufficient condition of Theorem 2.

Theorem 2 *The river routing instance defined above is routable in $l \ge 1$ layers (each layer routing whole nets) using at most $t \ge 0$ tracks per layer if and only if*

(a) $a_{i+lt} - b_i \ge t$

(b) $b_{i+lt} - a_i \ge t$

for every $i \le m - lt$.

Proof: First, we establish that (a) and (b) are necessary for l layer routing. Since the proofs for (a) and (b) are similar, we provide that for (a) only. Suppose that $a_{i+lt} - b_i < t$ for some i. Consider the $lt + 1$ terminal pairs (A_j, B_j), $i \le j \le i + lt$. When routing these on l layers,

```
procedure RoundRobin ;
{ Assign the $m$ nets to $l$ layers. }
begin
    for $i := 1$ to $m$ do
        assign net $(A_i, B_i)$ to layer $(i \bmod l) + 1$ ;
end ;
```

Figure 3: Round robin layer assignment

at least one layer has to be assigned $\geq t+1$ terminal pairs. So, suppose that terminal pairs (A'_1, B'_1), (A'_2, B'_2), $\ldots$, (A'_{t+1}, B'_{t+1}), $\ldots$ are assigned to the same layer for river routing. We may assume that $a'_1 < a'_2 < \ldots < a'_{t+1}$ and $b'_1 < b'_2 < \ldots < b'_{t+1}$. Since $a'_{t+1} \leq a_{i+lt}$ and $b'_1 \geq b_i$, $a'_{t+1} - b'_1 \leq a_{i+lt} - b_i < t$. From Theorem 1, it follows that the terminal pairs (A'_j, B'_j), $1 \leq j \leq t+1$, cannot be river routed on a single layer. Hence, (A_j, B_j), $i \leq j \leq i+lt$, cannot be river routed on l layers. So, (A_j, B_j), $1 \leq j \leq m$, cannot be river routed on l layers. As a result, (a) is a necessary condition.

To show that (a) and (b) are sufficient conditions for routability, we present two algorithms (RoundRobin and Greedy) that assign the nets to layers in such a way that each layer is river routable when both (a) and (b) are satisfied. The correctness of these algorithms is established in Theorems 3 and 4, respectively. □

Theorem 3 *The layer assignment produced by the RoundRobin procedure is river routable if*

(a) $a_{i+lt} - b_i \geq t$

(b) $b_{i+lt} - a_i \geq t$

for all $i \leq m - lt$.

Proof: Let (A'_1, B'_1), (A'_2, B'_2), $\ldots = (A_j, B_j)$, (A_{j+l}, B_{j+l}), (A_{j+2l}, B_{j+2l}), $\ldots$ be the nets assigned to layer $(j \bmod l) + 1$, $j \leq l$. So, $a'_i = a_{j+(i-1)l}$ and $b'_i = b_{j+(i-1)l}$. Hence,

$$\begin{aligned} a'_{i+t} - b'_i &= a_{j+(i+t-1)l} - b_{j+(i-1)l} \\ &= a_{j+(i-1)l+tl} - b_{j+(i-1)l} \\ &\geq t \quad \text{(from (a))} \end{aligned}$$

Similarly, $b'_{i+t} - a'_i \geq t$. So, the layer assignment satisfies the conditions of Theorem 1 and is river routable using t tracks. □

```
procedure Greedy ;
{ Assign the $m$ nets to $l$ layers. }
begin
    for $i := 1$ to $m$ do
        assign net $(A_i, B_i)$ to layer $q$ such that $q$ is the smallest integer $\leq l$
        for which the conditions of Theorem 1 are not violated on layer $q$
        (if there is no such $q$, then fail) ;
end ;
```

Figure 4: Greedy layer assignment

Theorem 4 *If (a) $a_{i+lt} - b_i \geq t$ and (b) $b_{i+lt} - a_i \geq t$ for all $i \leq m - lt$, then procedure Greedy assigns nets to layers such that the assignment to each layer is routable using t tracks.*

Proof: If procedure Greedy is able to assign each of the m nets to a layer, then the layer assignments satisfy the conditions of Theorem 1 and so are routable using t tracks. Suppose the algorithm fails while trying to assign net (A_r, B_r) to a layer. At this time nets (A_i, B_i), $1 \leq i < r$, have been assigned to layers so as to satisfy the conditions of Theorem 1 and the assignment of net (A_r, B_r) to each of these layers violates these conditions. Consider first those layers, L_a, on which condition (a) is violated. For a layer $s \in L_a$, suppose that the assigned nets are $\ldots$, (A'_{j-t}, B'_{j-t}), $\ldots$, (A'_{j-1}, B'_{j-1}). Let $(A'_j, B'_j) = (A_r, B_r)$. Since $s \in L_a$, we have $a'_j - b'_{j-t} < t$. Now, if $b_{r-lt} \geq b'_{j-t}$, then $a_r - b_{r-lt} < t$ which violates condition (a) of this theorem. So, $b_{r-lt} < b'_{j-t}$. Since, $b_{r-lt} < b'_{j-t} < \ldots < b'_{j-2} < b'_{j-1}$, t of the $lt - 1$ nets (A_{r-lt+1}, B_{r-lt+1}), $\ldots$, (A_{r-1}, B_{r-1}) have been assigned to layer $s \in L_a$. Consequently, the layers in L_a account for $t|L_a|$ of these $lt - 1$ nets.

In a similar way, we can show that the remaining $l - |L_a|$ layers account for another $t(l - |L_a|)$ of these nets. This gives us a total of tl nets, whereas we had only $tl - 1$. This contradiction implies that procedure Greedy cannot fail unless conditions (a) and (b) are not satisfied. □

Procedure RoundRobin is easily seen to have complexity of O(m). A straightforward implementation of procedure Greedy will have complexity of O(ml). However, by using priority search trees [MCCR85] the complexity can be reduced to O$(m \log l)$. In practice, since l is quite small, it is unlikely that the priority search tree implementation will run faster than the straightforward implementation in which the l layers are checked in sequence. The

```
procedure MinimizeTracks ;  {or MinimizeLayers}
{ Determine the minimum number of tracks per layer (or minimum number
  of layers) needed for multilayer river routing }
begin
    $t := 0$ ;  {or $l := 1$}
    $i := 1$ ;
    while $(i \leq m - lt)$ do
        if $(a_{i+lt} - b_i < t)$ or $(b_{i+lt} - a_i < t)$
        then $t := t + 1$  {or $l := l + 1$}
        else $i := i + 1$ ;
end ;
```

Figure 5: Minimizing the number of tracks or layers

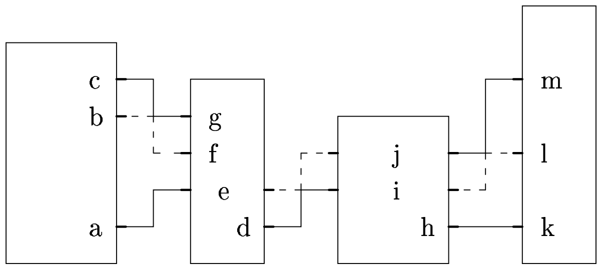
actual routing for all l layers can be done in O(mt) time using the computed layer assignment and the single layer routing algorithm of [LEIS83].

Using Theorem 2, we can develop a linear time algorithm to determine the minimum number of tracks needed to route an instance in l layers as well as to find the minimum number of layers needed for a t track routing. The algorithm for the former is given in Figure 5. This figure also shows the changes needed in case t is given and we wish to determine the minimum number of layers. The correctness of the algorithm follows from that of Theorem 2 and the fact that t (or l) is increased only if the current t (l) is found to be infeasible. The complexity is O(m) as neither i nor t (l) can exceed m. So, neither clause of the **if** statement can be executed more than $m - 1$ times.

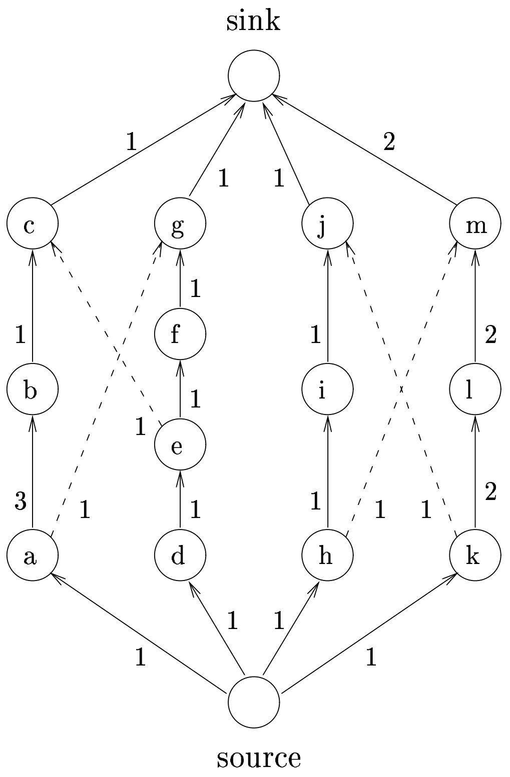
The developments of this section allow us to trivially extend all the results of [LIM93] to the case of multilayer joining of compacted cells. So, the multilayer minimum area join of two compacted cells with m nets can be obtained in O(m^2) time. If we wish to minimize the maximum wirelength while keeping area minimum, the asymptotic time complexity is still O(m^2). The total wire length can be minimized while keeping area minimum in O($m^2 \log m$) time.

3 Constraint Graph Representation

Lim [LIM92] has proposed the use of a constraint graph to determine the terminal positions in a row of compacted cells. This is for the case when the number of tracks in each routing



(a) Cell joining with 2 layers



(b) Constraint graph representatione

Figure 6: Constraint graph representation

channel is given and we wish to minimize the layout height. In the constraint graph, each cell is represented by a directed chain of vertices. Each cell terminal is represented by a vertex. The exception is when a compacted cell has terminals at the same y-position on both sides of the cell. In this case, the two terminals at the same y-position are represented by a single vertex. The vertex chain is linked in the direction of increasing y-position. The chain edges are labeled by the minimum allowable terminal separation. In addition, the constraint graph contains a source vertex that represents the bottom of the layout and a sink vertex that represents the layout top. The source vertex connects to the bottom of each chain and the top of each chain is connected to the sink vertex.

Figure 6(b) shows the chains (solid edges) for the four cell row of Figure 6(a). To complete the constraint graph directed edges are added to introduce the channel routing constraints of Theorem 2. These are represented by the broken edges of Figure 6(b). Figure 6(b) is for the two layer case.

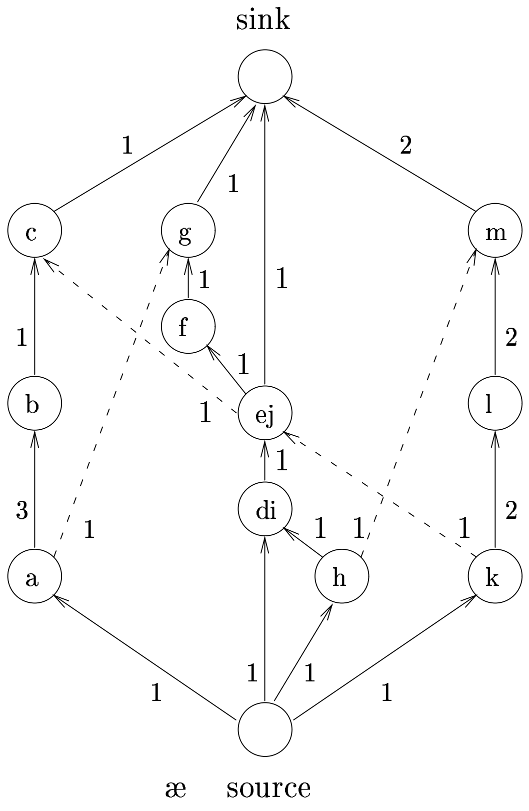


Figure 7: Merge in constraint graph

Lim [LIM92] has shown that the constraint graph is acyclic provided the number of tracks in each routing channel is > 0. He has proposed handling channels with zero tracks by finding first the minimum area joining of the adjacent cells (only cell stretching is permitted now) and then combining these two cells into one. I.e., the two cells are replaced by their minimum area join. This strategy can be shown to result in non optimality of the algorithm proposed in [LIM92]. To preserve optimality, it is necessary to merge the vertices that represent terminals that are the endpoints of nets that are to be routed using no tracks as in Figure 7. The resultant constraint graph is also acyclic.

It is easy to see that the number of vertices and edges in the constraint graph is $O(n)$ where n is the total number of terminals. Furthermore, the graph can be constructed in $O(n)$ time given the number of routing layers and the number of tracks in each channel. The constraint graph described by us is identical to that of [LIM92] except in the way channels with zero tracks are handled and in that our graph is defined for $l \geq 1$ routing layers while

```
procedure Heuristic1 ;
begin
    for $i := 1$ to $c - 1$ do
    begin
        determine the minimum area join of each pair of adjacent cells ;
        select the pair that has minimum area and replace it with its
        minimum area join ;
    end ;
end ;
```

Figure 8: Heuristic 1

that of [LIM92] is only for $l = 1$.

The length of the longest path from the source vertex of the constraint graph to each of the remaining vertices can be computed in $O(n)$ time by doing this in topological order [HORO94, Section 6.5]. It is easy to see that if each terminal is placed at a vertical position given by the longest path length from the source, then all nets can be routed in the given number of tracks (as the conditions of Theorem 2 are satisfied in each routing channel). Furthermore, Lim [LIM92] has shown that such a positioning of terminals results in a stretched layout of minimum height for the given channel widths. As a result, when channel widths are known, cells can be stretched to minimize area in $O(n)$ time. The channel widths that result in minimum area can be determined in $O(n(n/c)^{c-1})$ time where c is the number of cells by trying out all possible channel widths [LIM92]. Since this is feasible only for small c, we propose several heuristics in the next section.

4 Heuristics to Minimize Area

We formulate three greedy heuristics to obtain the minimum area join of a row of c compacted cells that have a total of n terminals.

4.1 Heuristic 1

The heuristic is described in Figure 8.

In each iteration of the **for** loop we examine every pair of adjacent cells. For each pair, the minimum area join is found using the algorithm of [LIM93] extended to the multilayer case as discussed in Section 2. The pair which has the minimum area join is replaced by a single cell that represents this join. So following each iteration of the **for** loop the number of cells decreases by one. When the **for** loop terminates, we are left with a single cell that represents the join of all c cells. The time needed to determine the minimum area join of a pair of cells with n_i nets between them is $\mathrm{O}(n_i^2)$. The time to do this for all pairs of adjacent cells is $\mathrm{O}(\sum_i n_i^2) = \mathrm{O}(n^2)$. So, the **for** loop iteration with $i = 1$ takes $\mathrm{O}(n^2)$ time. On subsequent iterations, only the two pairs that include the cell introduced in the previous iteration need to have their minimum area join computed. Since each cell pair being considered includes at least one composite cell, the minimum area join is computed by considering the portion of the constraint graph that represents all the basic cells in the cell pair. Channel widths for channels within a composite cell are not changed while obtaining the minimum area join of the cell pair. However as different channel widths for the channel between the two (composite) cells being joined are tried, the constraint graph is used to determine the minimum height of the combined cell. So, the time to combine two (composite) cells with n_i terminals in the channel between them is $\mathrm{O}(nn_i)$. Hence the time for the remaining $c-2$ iterations is $\mathrm{O}(n\sum_i n_i) = \mathrm{O}(n^2)$. The overall complexity of Heuristic 1 is therefore $\mathrm{O}(n^2)$. In case the terminals are uniformly distributed over the cells, $n_i = \mathrm{O}(n/c)$ for all i. The time for the first iteration of the **for** loop is now $\mathrm{O}(n^2/c)$ and that for each of the remaining iterations is $\mathrm{O}(n^2/c)$. The overall time is $\mathrm{O}(n^2)$.

4.2 Heuristic 2

In this heuristic, we begin by assigning each channel the number of tracks needed to route the channel with no cell stretching. This number can be determined in $\mathrm{O}(n_i)$ time for a channel with n_i nets as described in Section 2. The time taken to do this for all $c-1$ channels is $\mathrm{O}(n)$. The configuration obtained in this way is the maximum width layout. Starting from this configuration, we reduce the total number of tracks available across all $c-1$ channels by one on each iteration. For this, the effect of a one track reduction is computed for each channel. The minimum layout height is determined by computing the length of the longest path in the constraint graph of Section 3. The track reduction is done in the channel that results in the smallest layout height (hence the minimum area for the given number of tracks). The algorithm is stated more formally in Figure 9.

When the algorithm terminates, A is the area of the minimum area join found by the

```
**procedure** Heuristic2 ;
**begin**
    for each channel determine the number of tracks, $t_i$,
    needed to route with no stretching, $1 \leq i < c$ ;
    $t := \sum_{i=1}^{c-1} t_i$ ;
    set up the constraint graph using $t_i$ tracks in channel $i$, $1 \leq i < c$ ;
    compute layout area, $A$ ;
    **for** $tracks := t$ **downto** 1 **do**  {reduce by 1}
    **begin**
        **for** $i := 1$ **to** $c - 1$ **do**
        **begin**
            reduce the number of tracks in channel $i$ by 1 ;
            modify the constraint graph to reflect this ;
            determine the length of the longest path in the graph
            and from this the layout area, $a_i$ ;
        **end** ;
        select $j$ such that $a_j = min\{ a_i \}$ ;
        reduce the number of tracks in channel $j$ by 1 ;
        $A = min\{ A, a_j \}$ ;
    **end** ;
**end** ;
```

Figure 9: Heuristic 2

heuristic. To reconstruct the layout, it is necessary to store the tracks per channel each time A is updated in the statement $A = min\{ A, a_j \}$.

For the time complexity, we see that the steps that precede the outer **for** loop take O(n) time. Each iteration of the outer loop takes O(nc) time. Hence this loop contributes a total of O(nct) to the time. Since $t =$ O(n), the overall time complexity of Heuristic 2 is O(n^2c).

4.3 Heuristic 3

Unlike Heuristic 2 which attempts to minimize the layout height for each value of t, the total number of tracks, Heuristic 3 attempts to minimize the width (i.e., total number of tracks) for each choice of layout height. The heuristic begins with a layout height, ht, equal to the height of the tallest compacted cell. At each iteration, the next layout height to use is computed as described later. During each iteration, cells are combined in groups of at most k ($k > 1$ is a parameter to the heuristic). Each group of combined cells is replaced by its minimum area join subject to the constraint that the height of the join does not exceed

```
procedure Heuristic3 ;
begin
    $ht$ := height of the tallest cell ;
    repeat { minimize width subject to height $\leq ht$ }
        repeat { do this by combining $k$ cells at a time }
            select $k$ adjacent cells such that the minimum height cell is selected
            and the height of the tallest selected cell is minimum
            (if there are fewer than $k$ cells, then select all of them) ;
            obtain the minimum area layout for the selected cells under
            the constraint that the layout height does not exceed $ht$ ;
            during the preceding step record the next value of $ht$
            that is possible for a layout ;
        until one cell remains ;
        compute the area of the remaining cell and record it
        if it is less than the minimum area found so far ;
        if there is no next height then terminate ;
        $ht$ := next height ;
    until false;
end ;
```

Figure 10: Heuristic 3

ht. This joining of $\leq k$ cells at a time continues until only one cell remains. Its area is computed and recorded. The minimum area obtained over all heights tried is then reported as the best. Heuristic 3 is given in Figure 10.

In our implementation of heuristic 3, the minimum area join of k cells is found by considering the portion of the constraint graph for all the basic cells included in these k cells. So, for this purpose composite cells are not handled as single cells. Rather, as in Heuristic 1, the basic cells they are composed of are considered and channel widths previously assigned to the associated channels are not changed. Track assignment is done only for the $k-1$ channels between the k composite cells. We found this to give better results than when composite cells were regarded as atomic. For the case $k = 2$, the minimum area is determined by a binary search over the number of tracks in the single channel. This takes $O(n \log n_i)$ time where n_i is the number of nets in channel i. Thus the time needed for the inner **repeat** loop when $k = 2$ is $O(cn \log n)$ (for uniform terminal distribution it is $O(cn \log(n/c))$. During the binary search, the heights corresponding to channel widths that require height $> ht$ are recorded. The minimum of these heights yields the next value of ht.

When $k > 2$, all track combinations for the $k-1$ channels are tried as in Section 3.

Again, each composite cell is broken up into its basic cells. As different track combinations are tried, we record the minimum height $> ht$ that results from any track combination. This gives the next value of ht. The time for the inner **repeat** loop is $O((c/(k-1))n(n/k)^{k-1})$ (or $O((c/(k-1))n(n/c)^{k-1})$ when terminals are uniformly distributed).

In all our experiments, the outer repeat loop was iterated fewer than $(k-1)n$ times. To ensure that the number of iterations is $O(kn)$, one may adopt the following scheme. When the number of iterations first reaches $(k-1)n$, compute a set of at most n new heights by beginning with the current constraint graph. This uses the current assignment of number of tracks in each channel. Heuristic 2 is next used to reduce the total number of available tracks by one and determine the height needed to complete the routing with the reduced number of tracks. This process gives us at most n new heights $h_1 < h_2 < \ldots < h_p$. Heuristic 3 is now resumed with h_1 as the next height. Only two iterations are performed. Then Heuristic 3 is resumed with $max\{\ h_2, ht\ \}$ as the next height. Again two iterations of the outer **repeat** loop are done. Next the heuristic is resumed with $max\{\ h_3, ht\ \}$ as the next height. This continues until we have gone through p resumptions of the heuristic. With this scheme to limit the number of iterations, the complexity of Heuristic 3 becomes $O(cn^2 \log n)$ when $k = 2$ and $O((c/(k-1))kn^2(n/k)^{k-1}) = O(cn^2(n/k)^{k-1})$ when $k > 2$. For the case when the n terminals are uniformly distributed over the c cells, the complexity is $O(cn^2 \log(n/c))$ when $k = 2$ and $O((c/(k-1))kn(n/c)^{k-1}) = O(cn(n/c)^{k-1})$ when $k > 2$. One may verify that since Heuristic 3 tries the maximum useful height (i.e., the height needed when no routing tracks are available), it generates optimal solutions when $k = c$.

5 Experimental Results

We programmed our three heuristics as well as the heuristic Fang [CHEN89] in C and ran tests on a single KSR processor. Optimal solutions for instances with upto nine cells were obtained using the corrected version of the exhaustive search algorithm of [LIM92]. Our test set consisted of instances that had a number of cells, c, equal to one of the numbers in the set {3, ..., 9, 10, 20, 50, 100}. For each value of c, there were twenty instances and the results were averaged over these instances. An instance with c cells had $c-1$ routing channels. The number, t, of terminals on either side of each routing channel was equal to c

for $3 \leq c \leq 9$ and was 10 for the other values of c. In addition, when $c = 100$, we also had instances with 20 terminals on either side. In our experiments, we considered only single layer and two layer routing.

Table 1 gives the average percentage by which the area of the single layer solutions generated by each of the heuristics exceeded the area of the single layer optimal solution. As is evident, each of the heuristics proposed in this paper gave noticeably better solutions than did Fang. This table is only for the cases $3 \leq c \leq 9$ as for $c > 9$ the optimal algorithm of [LIM92] required too much time to complete.

cells	t	Fang	Heuristic 1	Heuristic 2	Heuristic3		
					$k=2$	$k=3$	$k=4$
3	3	5.9	0.5	0.2	0	–	–
4	4	10.0	0.9	0.1	0	0	–
5	5	11.0	3.7	0.3	0.3	0.1	0.1
6	6	12.7	2.4	0.3	0.2	0.2	0.0
7	7	16.1	3.5	0.3	0.1	0.1	0.1
8	8	17.9	2.7	0.4	0.3	0.3	0.1
9	9	18.8	3.5	0.3	0.4	0.3	0.1

t = number of terminals on each side of each routing channel

Table 1: Error rate (%) over optimal, $l = 1$

In table 2, we have used the single layer solution produced by Fang as the benchmark against which the solutions obtained by our three heuristics are compared. This table gives the average percentage by which the area of the solutions produced by our heuristics is less than that of the solutions produced by Fang. Our solutions have area 9 to 18% less.

Table 3 compares the computing time requirements of the various algorithms for the case of one layer. The optimal algorithm is useful only for small values of c (say upto 7). While Fang is significantly faster than the heuristics proposed here, the quality of the solutions generated by our heuristics is superior.

Table 4 is the analog of table 1 for the case of two layers. Again, our heuristics performed considerably better than did Fang. Table 5 gives the improvement in area due to increasing the number of routing layers from one to two. This is influenced somewhat by the width of cells which in our case ranged from 5 to 30 times the track separation. With narrower cells,

cells	t	Heuristic 1	Heuristic 2	Heuristic3		
				$k = 2$	$k = 3$	$k = 4$
10	10	9.4	14.0	14.0	14.2	14.3
20	10	10.5	15.4	15.4	15.6	15.6
50	10	10.6	16.1	16.1	16.2	–
100	10	9.1	16.5	16.5	16.6	–
100	20	9.3	18.3	18.0	–	–

Table 2: Improvement (%) over Fang, $l = 1$

the impact of the second layer would have been greater and with wider cells, it would have been less. Also, the impact of the second layer is more when more routing tracks are needed. For the smaller instances of table 4, for example, the optimal solutions with $l = 2$ required, on average, only 1.8% less area than when $l = 1$.

Table 6 is the analog of table 2 for the case of two layers. The results are similar to those in table 2. Table 7 gives the average computing times for the two layer instances. These are less than for the one layer case as the constraint graph has fewer edges.

For large c, we recommend the use of heuristic 2 or 3 (with $k = 2$) and for small c we recommend using heuristic 3 (with $k = 3$ or 4).

6 Conclusion

We have developed necessary and sufficient conditions for multilayer river-routing. In addition, two simple algorithms to do multilayer river routing using the fewest number of tracks were developed. Next, we considered the problem of joining a row of compacted cells and developed heuristics to stretch cells and river-route the nets so that the layout area is minimized. Our proposed heuristic was compared, experimentally, with Fang [CHEN89] and found to produce layouts with less area. However Fang is faster. We recommend the use of our Heuristic 3 with $k = 3$ or 4 in practice.

cells	t	Fang	Heuristic 1	Heuristic 2	Heuristic3			Optimal
					$k=2$	$k=3$	$k=4$	
3	3	0.0	0.01	0.02	0.02	–	–	0.01
4	4	0.0	0.02	0.02	0.04	0.09	–	0.05
5	5	0.0	0.03	0.04	0.09	0.36	1.2	0.77
6	6	0.0	0.03	0.08	0.21	0.66	3.7	13
7	7	0.0	0.04	0.14	0.50	3.4	27	278
8	8	0.0	0.07	0.24	0.84	4.4	34	$1.8^{\dagger}$
9	9	0.0	0.09	0.42	1.8	16	72.3	$47^{\dagger}$
10	10	0.0	0.14	0.78	3.1	19	334	–
20	10	0.01	0.32	5.8	16	117	1196	–
50	10	0.01	1.4	93	127	896	–	–
100	10	0.03	4.4	782	590	4873	–	–
100	20	0.06	11	3022	4087	–	–	–

Times are in seconds.
$\dagger$: Times are in hours.

Table 3: Time taken, $l = 1$

References

[BOYE88] Boyer, D. G., Symbolic Layout Compaction Review, 25th Design Automation Conference, June 1988, pp. 383-389.

[CHEN89] Cheng, G and A. Despain, Fang: A Joiner for Compacted Cells, in VLSI 89, 1989, pp. 455-463.

[HORO94] Horowitz, E. and S. Sahni, Fundatamentals of Data Structures in Pascal, 4th Edition, W. H. Freeman and Company, 1994.

[LEIS83] Leiserson, C. E. and R. Y. Pinter, Optimal Placement for River Routing, SIAM Journal on Computing, 1983, pp. 447-462.

[LIM92] Lim, A., Efficient Algorithms for CAD in VLSI, PhD Dissertation, Univ. of Minnesota, 1992.

[LIM93] Lim, A., S. Cheng, and S. Sahni, Optimal Joining of Compacted Cells, IEEE Transactions on Computer, Vol 42, No 5, May 1993, pp. 597-607.

cells	t	Fang	Heuristic 1	Heuristic 2	Heuristic3		
					$k=2$	$k=3$	$k=4$
3	3	6.3	0.5	0	0	–	–
4	4	9.8	1.1	0.1	0	0	–
5	5	11.0	2.7	0.1	0.2	0.1	0.0
6	6	13.0	1.4	0.1	0.0	0.0	0.1
7	7	15.9	2.3	0.1	0	0	0
8	8	16.8	1.7	0.2	0	0.1	0.0
9	9	18.4	2.2	0.1	0.1	0.1	0.0

Table 4: Error rate (%) over optimal, $l = 2$

cells	t	Fang	Heuristic 1	Heuristic 2	Heuristic3		
					$k=2$	$k=3$	$k=4$
10	10	4.1	4.7	3.3	3.3	3.0	3.0
20	10	4.1	5.8	3.4	3.4	3.3	3.2
50	10	4.6	5.2	3.4	3.2	3.2	–
100	10	4.7	6.3	3.5	3.4	3.4	–
100	20	7.1	10.4	4.8	5.2	–	–

Table 5: Improvement (%) over $l = 1$ cases

[MCCR85] McCreight, E. M., Priority Search Trees, SIAM J. Comput., Vol 14, No 2, May 1985, pp. 257-276.

[MIRZ87] Mirzaian, A., River Routing in VLSI, Journal of Computer and System Sciences, Vol 34, 1987, pp. 43-54.

[PINT82] Pinter, R. Y., On routing two-point nets across a channel, 19th Design Automation Conference, June 1982, pp. 894-902.

[PINT83] Pinter, R. Y., River Routing: Methodology and Analysis, Third Caltech Conference on VLSI, March 1983.

[WEST81] Weste, N., Virtual Grid Symbolic Layout, 18th Design Automation Conference, June 1981, pp. 225-233.

cells	t	Heuristic 1	Heuristic 2	Heuristic3		
				$k = 2$	$k = 3$	$k = 4$
10	10	10.0	13.3	13.2	13.2	13.3
20	10	12.2	14.8	14.8	14.9	14.8
50	10	11.2	15.1	14.9	15.0	–
100	10	10.7	15.4	15.4	15.4	–
100	20	12.5	16.3	16.3	–	–

Table 6: Improvement (%) over Fang, $l = 2$

cells	t	Fang	Heuristic 1	Heuristic 2	Heuristic3			Optimal
					$k = 2$	$k = 3$	$k = 4$	
3	3	0.0	0.01	0.02	0.02	–	–	0.0
4	4	0.0	0.02	0.02	0.03	0.06	–	0.04
5	5	0.0	0.02	0.02	0.07	0.25	0.75	0.60
6	6	0.0	0.02	0.04	0.13	0.38	2.0	11
7	7	0.0	0.04	0.06	0.29	1.9	14	223
8	8	0.0	0.06	0.12	0.53	2.5	19	$1.5^{\dagger}$
9	9	0.0	0.06	0.20	0.96	7.5	34	$39^{\dagger}$
10	10	0.01	0.09	0.32	1.7	9.9	153	–
20	10	0.01	0.24	2.7	9.9	65	651	–
50	10	0.01	1.1	42	85	586	–	–
100	10	0.02	3.7	357	472	3069	–	–
100	20	0.05	7.9	1377	3166	–	–	–

Times are in seconds.
$\dagger$: Times are in hours.

Table 7: Time taken, $l = 2$