FEASIBLE OFFSET AND OPTIMAL OFFSET FOR GENERAL SINGLE-LAYER CHANNEL ROUTING*

RONALD I. GREENBERG† AND JAU-DER SHIH‡

Abstract. This paper provides an efficient method to find all feasible offsets for a given separation in a VLSI channel routing problem in one layer. The prior literature considers this task only for problems with no single-sided nets. When single-sided nets are included, the worst-case solution time increases from $\Theta(n)$ to $\Omega(n^2)$, where n is the number of nets. But if the number of columns c is $O(n)$, the problem can be solved in time $O(n^{1.5}\lg n)$, which improves upon a "naive" $O(cn)$ approach. As a corollary of this result, the same time bound suffices to find the optimal offset (the one that minimizes separation). Better running times result when there are no two-sided nets or all single-sided nets are on one side of the channel. This paper also gives improvements upon the naive approach for $c \neq O(n)$, including an algorithm with running time independent of c. An interesting algorithmic aspect of the paper is a connection to discrete convolution.

Key words. VLSI layout, channel routing, single-layer wire routing, discrete convolution, combinatorial algorithms

AMS subject classifications. 68Q35, 68Q25

1. Introduction. Much attention has been given to planar or single-layer wire routing for VLSI chips. Most popular has been river routing in the restricted sense of the term, the connection of two parallel rows of corresponding points[1], e.g., [11] and the references therein. Other works have considered routing within a rectangle [2], placement and routing within a ring of pads [1], or routing between very general arrangements of modules [10, 4].

Ironically, single-layer routing may become more relevant as technology evolves towards chips with increasing numbers of layers. With many layers, it becomes more likely that an individual layer can be dedicated to a coplanar subset of the original collection of nets. For example, the heuristic multi-layer channel router MulCh [7] improved upon previous multilayer channel routers by dividing the problem into essentially independent subproblems of one, two, or three layers.

In this paper, we consider the single-layer channel routing problem, which is more general than the more heavily studied river routing problem. Channel routing is similar to river routing in that both deal with the interconnection of terminals lying in two parallel rows (sides of the channel); also, for simplicity, we restrict attention to 2-point nets as in river routing.[2] But we allow nets that have both their terminals on the same side of the channel, contrary to river routing. The existence of these *single-sided* nets is both realistic (as in the example problems of [7]) and a significant algorithmic complication. As shown in Figure 1, the usual convention is to draw the rows of terminals horizontally; only the region between these rows is available for routing. We refer to single-sided nets that have their two terminals on the top

* This work was supported in part by NSF grant CCR-9109550.

† Electrical Engineering Department, University of Maryland, College Park, MD 20742 (rig@eng.umd.edu)

‡ Department of Information Engineering, Kaohsiung Polytechnic Institute, Ta-Hsu, Kaohsiung, Taiwan 84008, Republic of China (jdshih@nas04.kpi.edu.tw)

1 This is the only usage of the term "river routing" in this paper; we refer to more complicated variations of the problem as "single-layer" or "planar" routing.

2 Multiterminal nets can be handled by a transformation that might be considered "folklore". It is described in [8] in the context of showing that minimum separation problems can be solved even more easily than by actually applying the transformation.

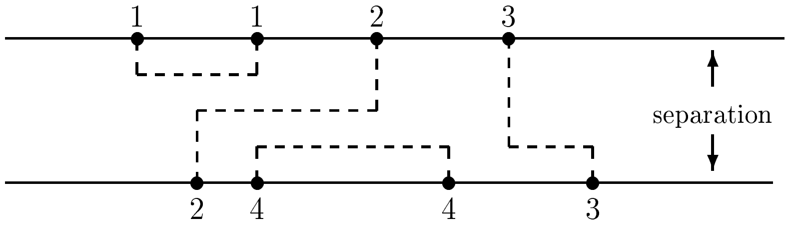

FIG. 1. *An example of a routed single-layer channel.*

(bottom) as *upper (lower) nets*. Nets with terminals on opposite sides are referred to as *two-sided* nets. We restrict attention to a rectilinear, grid-based model in which terminals lie on gridpoints and wires are disjoint paths through grid edges. We use c to denote the total number of grid columns from the leftmost terminal to the rightmost terminal and n to denote the number of nets.

The greatest attention has been given to the *minimum separation* version of the problem. In this case, we assume that the horizontal positions of the terminals are completely fixed, but we seek the minimum vertical distance between the two rows of terminals that allows the routing to be completed. An $O(n)$ time solution in the river routing case was given in [6]. Though some erroneous solutions have been published for the general channel routing case, a simple and correct $O(n)$ algorithm is provided in [8].

In this paper, other important versions of the river routing problem are solved in the context of channel routing; in these problems, we allow the rows of terminals to be offset relative to one another. That is, we allow the upper row of terminals to be slid as a block to the left or right, though individual terminals do not shift position relative to one another. (This models the situation in which we are trying to wire together two modules each having terminals on one side, and we have substantial freedom on how to place the modules.) The *optimal offset problem* involves finding the offset that minimizes the amount of separation necessary to route the problem. A related problem, which we refer to as the *feasible offset problem* is to determine all offsets that are feasible (i.e., give enough room to route) at a given separation. In the river routing context, the second problem is usually called the *offset range problem*, since the feasible offsets always constitute a single continuous range, but this property does not hold for channels with single-sided nets.

Mirzaian [11] showed that feasible offset and optimal offset can be computed in $O(n)$ time in the river routing case, but we are not aware of any published solutions for channels with single-sided nets. One complication that arises when single-sided nets are included is that the solution time is no longer insensitive to the number of columns in the problem (at least for feasible offset). As illustrated in Figure 2, if the number of columns is large, the number of disjoint intervals of feasible offsets may be $\Omega(n^2)$. But if $c = O(n)$, we show that feasible offset can be solved in $O(n^{1.5} \lg n)$ time. This improves on the naive $O(cn)$ time obtained by running the $O(n)$ algorithm for the minimum separation problem at each of the $2c$ offsets that may need to be checked. In the remainder of this paper, we express our running times in terms of c as well as n where necessary, but we concentrate on obtaining a good running time when $c = O(n)$. Later, we give an algorithm that is less efficient for $c = O(n)$ but has a running time independent of c.

The remainder of this paper is organized as follows. In Section 2, we introduce

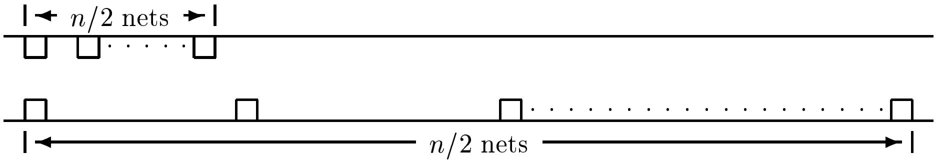

FIG. 2. *For small separation, the number of disjoint intervals of feasible offsets of the channel above is* $\Omega(n^2)$

some additional terminology and notation, and we show how to solve the feasible offset problem for a channel in which all nets are single-sided. In this case, the running time with $c = O(n)$ is $O(n^{1.5}\sqrt{\lg n})$, which leads to an $O(n^{1.5}\sqrt{\lg n})$ algorithm for optimal offset. (The optimal offset problem as defined above is trivial when all nets are single-sided; large offset minimizes separation. But we can handle a nontrivial generalization of the problem in which certain offsets are disallowed.) In Section 3, we show how to combine ideas from Section 2 with some new ideas to obtain solutions for channels with both single-sided and two-sided nets. For the general channel, the running time to solve either feasible offset or optimal offset is $O(n^{1.5}\lg n)$. Section 4 provides concluding remarks and some additional results. In particular, feasible offset and optimal offset can be solved in time $O(n^2 \lg n)$ independent of c. Also, the *optimal placement problem*, involving multiple modules on each side of the channel, can be handled in $O(n^3)$ time.

2. Channels with Single-Sided Nets Only. In this section, we deal with the special case of channels with only single-sided nets. Much of the work we do here will help us in the next section where we consider channels that have both single-sided and two-sided nets.

We begin by explaining some notation and terminology that we use throughout this paper. First, we use L, U, and T for the sets of lower, upper, and two-sided nets, respectively, and N for the complete set of nets in the channel. In addition, we often use the same notation interchangeably for a set of nets or for a lower or upper *contour*. The contour of a set of lower nets is the upper boundary of the routing region consumed in the routing of those nets that minimizes total wire length. That is, when the nets are routed as tightly as possible against the bottom of the channel, the contour is formed by the uppermost nets and portions of the channel boundary. The contour of a set of upper nets is defined similarly. We also refer often to subsets of contours, which simply means restricting the contour to certain columns (even though there may be no set of nets that would generate the resulting contour). We use the notations FOP and OOP to refer to the feasible offset problem and optimal offset problem, respectively. We also use the more precise notation FOP(s, A) to represent the set of solutions to the feasible offset problem with separation s and the set A of nets (or contours or contour fragments). We also use analogous notations SSFOP and SSOOP for the corresponding problems when all nets are single-sided. (For optimal offset, we permit the problem specification to disallow some set of offsets, e.g., all offsets $\geq c/2$; otherwise SSOOP is trivial.)

Our first step in solving SSFOP is to find the contours of the upper and lower nets. We use Pinter's result that $O(n)$ time suffices to find a contour (i.e., the coordinates of all the bends in the contour) [12].

LEMMA 2.1 (PINTER). *The contour of a set of n single-sided nets can be found in $O(n)$ time.* $\square$

Once we find the contours of the upper and lower nets, SSFOP can be expressed simply in terms of these contours. At each column, we define the *extension* of a contour to be the distance that the contour extends into the channel at that column. Then we are simply seeking all offsets for which no vertical cut corresponds to extensions of the upper and lower contours that sum to more than the separation under consideration. One way to solve this problem would be to compute the discrete convolution of the two sequences of extensions with the max and $+$ operators substituted for the usual $+$ and $\times$. It is unknown whether max, $+$ convolution for vectors of length n can be computed in better than $\Theta(n^2)$ time; still it will be seen that there is some relationship between convolution and our solution technique for SSFOP.

We begin with a general lemma allowing us to decompose SSFOP into smaller instances of the problem. In each of the smaller problems, we use only a portion of the lower contour, while retaining the entire upper contour. In fact, the lemma applies even when there are also two-sided nets. (Naturally, we also could switch the roles of the lower and upper contour.)

LEMMA 2.2. *Let $L_1, L_2, \ldots, L_k$ be any subsets of the contour L of the lower nets such that $L_1 \cup L_2 \cup \cdots \cup L_k = L$, and let A be an additional set of nets, then* $\mathrm{FOP}(s, L \cup A) = \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup A)$.

Proof. This follows from the fact that routing is possible if and only if each line segment from the top of the channel to the bottom of the channel is long enough (in the L_∞ metric) to accommodate the number of nets that must cross it (i.e., each *cut* is *safe*). More detail on the theory of single-layer routability can be found in [10]; see especially Section 2.1. □

We now proceed to decompose the lower contour into pieces that are easier to handle and are not too numerous. The next three lemmas are directed towards handling pieces of the contour that have large extension, and the following two lemmas handle portions of the contour in which there are not too many distinct extensions. Then we show how to put these two ideas together to solve the entire problem.

For the next lemma, we define a special type of contour fragment, such that if it comprises the entire lower contour, then SSFOP is particularly easy to solve. A *monotonic* subset of the lower contour L is a subset of L, such that the extensions within the selected columns are monotonically nondecreasing or monotonically nonincreasing as we move across the columns.

LEMMA 2.3. *If L_m is a monotonic subset of the lower contour, and U is the upper contour, then we can solve* $\mathrm{SSFOP}(s, L_m \cup U)$ *in $O(n)$ time.*

Proof. Without loss of generality, assume the (nonzero portion of the) lower contour has nondecreasing extensions from left to right. We need only march across the columns of the upper contour once from left to right. Initially, we consider a far left position for the lower contour (highly negative offset). The check for each column of the upper contour involves adding the upper extension to the lower extension for the corresponding column of the lower contour, based on the current offset, and comparing to the upper bound on separation. After any unsuccessful check, the current offset is incremented, and we do not yet advance to the next column of the upper contour. After each successful check, we move to the next column of the upper contour; prior columns never need to be rechecked at larger offsets since the lower contour is nondecreasing. When the rightmost column of the lower contour is involved in a successful check, a feasible offset has been found and, again, the current offset is incremented. The $O(c)$ approach just described can actually be improved to $O(n)$ time because of the following two facts. First, we really only need to look at columns of the upper contour where the upper contour bends. Second, there are at most n places where the extension of the lower contour changes, and preprocessing of the lower contour will allow us to increment offset sufficiently after each unsuccessful check so that we can proceed immediately to the next bend point of the upper contour. $\square$

In the next lemma, we show that not only are monotonic pieces of contour easy to handle, but that we don't have to check too many of them as long as we restrict attention to sections of contour with large extension. Here we define a monotonic subset to be *maximal* if no other monotonic subset contains it. Now we bound the number of maximal monotonic subsets in the portion of the contour with extension at least h.

LEMMA 2.4. *Let L_g be the subset of the lower contour containing only extensions greater than or equal to h, then L_g contains at most $c/2h$ maximal monotonic pieces.*

Proof. To have a maximal monotonic piece of the lower contour with extensions of at least h, there must be h lower nets nested one inside the next. Therefore, a maximal monotonic piece with extensions greater than or equal to h must span at least $2h$ columns, so L_g contains at most $c/2h$ maximal monotonic pieces. $\square$

We can now put together Lemmas 2.2, 2.3 and 2.4 to solve SSFOP efficiently for any piece of lower contour in which all extensions are large enough:

LEMMA 2.5. *If L_g is a subset of the lower contour containing only extensions greater than or equal to h, and U is the upper contour, then we can solve* $\mathrm{SSFOP}(s, L_g \cup U)$ *in $O(cn/h)$ time.*

Proof. By Lemma 2.2, we know that it suffices to solve the problem independently for each of the maximal monotonic pieces of L_g. By Lemma 2.4 there are $O(c/h)$ such pieces, and by Lemma 2.3, $O(n)$ time suffices for each piece. $\square$

What remains is to solve the SSFOP for the portion of the lower contour with small extensions. (Later, we'll show how to choose h, the dividing point between large and small extensions.) The next lemma handles the simplified case in which all extensions on the lower contour are 0 or 1. The following lemma goes on to handle h distinct extensions.

LEMMA 2.6. *If all extensions are* 0 *or* 1, *we can solve* SSFOP *in $O(c \lg c)$ time.*

Proof. The only interesting case is separation 1, and the feasible offsets correspond to the zero entries in the convolution of the upper and lower extensions. The convolution can be computed in $O(c \lg c)$ time by using the Fast Fourier Transform method. (See [5], for example.) $\square$

LEMMA 2.7. *If all extensions of the lower contour are at most h, we can solve* SSFOP *in $O(hc \lg c)$ time.*

Proof. From Lemma 2.2, $\mathrm{SSFOP}(s, L \cup U) = \bigcap_{i=1}^{h} \mathrm{SSFOP}(s, L_i \cup U)$, where L_i is the subset of the lower contour with extension i. We can now solve $\mathrm{SSFOP}(s, L_i \cup U)$ using Lemma 2.6 after assigning 1 to the lower extensions in L_i and those upper extensions exceeding $s - i$, and 0 to the other extensions. Since we have a total of h problems, each solvable in $O(c \lg c)$ time, the total time is $O(hc \lg c)$. $\square$

Now we can provide an overall solution to SSFOP by combining our results for contours with large extensions and contours with small extensions.

THEOREM 2.8. SSFOP *can be solved in $O(c\sqrt{n \lg c})$ time.*

Proof. $\mathrm{SSFOP}(s, L \cup U) = \mathrm{SSFOP}(s, L_l \cup U) \cap \mathrm{SSFOP}(s, L_g \cup U)$, where L_l is the subset of L with extensions less than h and L_g is the subset of L with extensions greater than or equal to h. Using Lemmas 2.7 and 2.5 with $h = \sqrt{n/\lg c}$, the solution time is $O(c\sqrt{n \lg c})$. $\square$

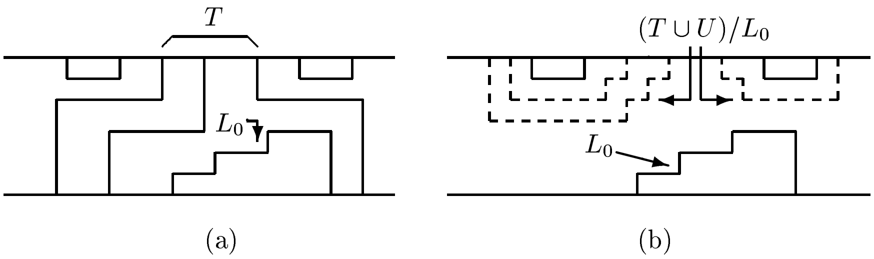

FIG. 3. *The effect of two-sided nets in (a) is incorporated into the top contour in (b). In this figure, L_0 is a monotonic portion of the lower contour.*

We can adapt the *halving* technique of [11] to solve SSOOP in the same time as SSFOP. The details will be shown in Section 3. The following theorem is just a simplified version of Theorem 3.8:

THEOREM 2.9. SSOOP *can be solved in* $O(c\sqrt{n \lg c})$ *time.* □

COROLLARY 2.10. SSFOP *and* SSOOP *can be solved in* $O(n^{1.5}\sqrt{\lg n})$ *time for* $c = O(n)$. □

3. General Single-Layer Channel. In this section, we use the ideas of section 2 to solve FOP and OOP when there are two-sided as well as single-sided nets. As before, we begin by computing the contours of the upper and lower nets. Also as before, we consider separately the portions of the lower contour with large extensions and the portions with small extensions and then show how to put these ideas together. But first we consider an intermediate case, when there are both single-sided and two-sided nets but all the single-sided nets are on one side.

LEMMA 3.1. *When all single-sided nets are on one side,* FOP *and* OOP *can be solved in* $O(n)$ *time.*

Proof. When the single-sided nets are on one side, we can extend the method of Mirzaian [11]. The basic idea is that as in river routing, the feasible offsets at a given separation form a continuous range whose bounds are determined by $O(n)$ *cut conditions.* More details can be found in [9]. □

To deal with the extra complications of two-sided nets, we also must introduce two new definitions.

First, let L_0 be a subset of the contour of the lower nets and T a set of two-sided nets whose lower terminals are to the left of L_0. Define T/L_0 as the upper contour obtained by pulling up the lower terminals of the nets in T and reconnecting them to the upper side to the left of preexisting terminals. That is, we convert the nets in T to single-sided nets without violating planarity and without moving what were their lower terminals to the wrong side of L_0. This notation is also used analogously for any set A of upper and two-sided nets as long as the lower terminal of each two-sided net is to the left or right of all nonzero extensions in L_0. In all cases, A/L_0 is the upper contour formed by moving lower terminals in T away from L_0 and to the upper side. Finally, the notation can also be used with a portion of the upper contour, in which case "upper" and "lower" are reversed throughout the definition. Figure 3 illustrates the definition of $(T \cup U)/L_0$.

For the second definition, let M be a subset of the contour of the upper or lower nets. We define $M|_s$ to be a new contour in which we replace all extensions exceeding $s - 1$ with extension $s - 1$.

We now proceed in the next two lemmas to handle a portion of lower contour with only large extensions. As before, the first lemma shows how to handle a monotonic piece of lower contour, and the second lemma handles a contour portion with large extensions by dividing it into maximal monotonic pieces.

LEMMA 3.2. *Let A be a set of upper and two-sided nets. Then we can solve* $\mathrm{FOP}(s, L_m \cup A)$ *in $O(n)$ time, where L_m is a monotonic portion of the lower contour.*

Proof. The solution is the intersection of the feasible offsets from two subproblems. In the first subproblem, we solve FOP without L_m (using Lemma 3.1). In the second subproblem, we retain L_m and we reroute the two-sided nets in the fashion shown in Figure 3, i.e., we determine $(U \cup T)/L_m$. Since we have already determined the infeasible offsets in the absence of L_m (in the first subproblem), however, we now ignore those portions of $(U \cup T)/L_m$ with extension exceeding $s-1$; we need only determine those offsets for which a vertical cut through $(U \cup T)/L_m$ *and* L_m has too large a sum of upper and lower extensions. So the second subproblem is just $\mathrm{SSFOP}(s, L_m \cup ((U \cup T)/L_m)|_s)$, which can be constructed and solved in $O(n)$ time by Lemmas 2.1 and 2.3. □

Now we combine Lemmas 2.2, 3.2 and 2.4 to solve FOP for a subset of L with large extensions. As before, we define large as exceeding h and specify the value of h later.

LEMMA 3.3. *If L_g is a subset of L containing only extensions greater than or equal to h, then we can solve* $\mathrm{FOP}(s, L_g \cup U \cup T)$ *in $O(cn/h)$ time.* □

Now that we have taken care of FOP with large extensions, we use the next two lemmas to deal with small extensions. The next lemma tells us how to transform certain instances of FOP into SSFOP, and will be used in handling general instances of FOP with small extensions.

LEMMA 3.4. *Let T_l and T_r be two sets of two-sided nets such that all the nets in T_l are to the left of those in T_r. (That is, the upper terminals in T_l are to the left of those in T_r and similarly for the lower terminals.) Also let U_l be a set of upper nets in which all terminals are to the left of (the upper terminals of) T_r, and L_r a set of lower nets in which all terminals are to the right of T_l. (See Figure 4.) Then,*

$$\begin{aligned}\mathrm{FOP}(s, U_l \cup T_l \cup T_r \cup L_r) \quad &= \quad \mathrm{FOP}(s, U_l \cup T_l \cup T_r) \cap \mathrm{FOP}(s, L_r \cup T_l \cup T_r) \cap \\ &\quad\quad \mathrm{FOP}(s, ((U_l \cup T_l)/L_r)|_s \cup ((L_r \cup T_r)/U_l)|_s) \ .\end{aligned}$$

Proof. The argument is similar to the one for lemma 3.2. At any given offset that is infeasible, either there is a vertical cut demonstrating infeasibility that goes through both U_l and L_r, or there is not. In the former case, we know that we can incorporate the effect of the two-sided nets into the upper and lower contours; i.e., solving $\mathrm{FOP}(s, ((U_l \cup T_l)/L_r)|_s \cup ((L_r \cup T_r)/U_l)|_s)$, as illustrated in Figure 4, will rule out the infeasible offsets of the first type. On the other hand, if the infeasibility does not result from interaction between U_l and L_r, it suffices to solve $\mathrm{FOP}(s, U_l \cup T_l \cup T_r)$ and $\mathrm{FOP}(s, L_r \cup T_l \cup T_r)$. Thus, intersection of the feasible offsets from these three problems provides the feasible offsets for the original problem. □

We can now solve FOP with small extensions:

LEMMA 3.5. *If the extensions of the upper and lower contours are all less than h, then we can solve* FOP *in $O(hc \lg^2 c)$ time.*

Proof. Let t be the number of two-sided nets. We first consider the case when $s < 4h$. Divide the channel into $t/4h$ blocks $B_1, B_2, \ldots, B_{t/4h}$, each spanning $4h$ two-sided nets as shown in Figure 5. Let L_i, U_i, and T_i denote the lower nets, upper nets, and two-sided nets in block i. (Single-sided nets at a boundary between blocks

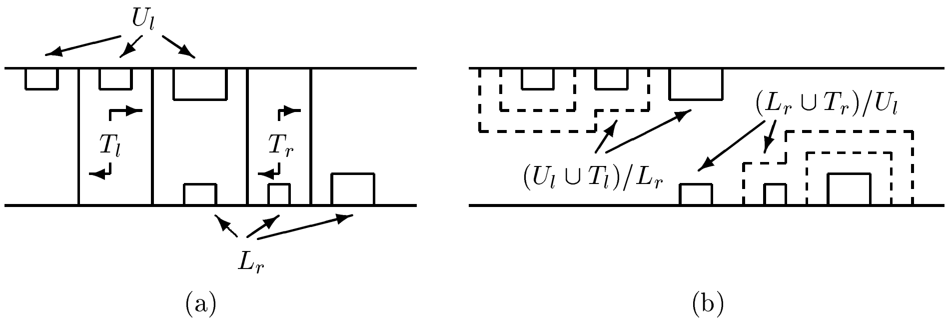

FIG. 4. *The effect of two-sided nets in (a) is incorporated into the upper and lower contours in (b).*

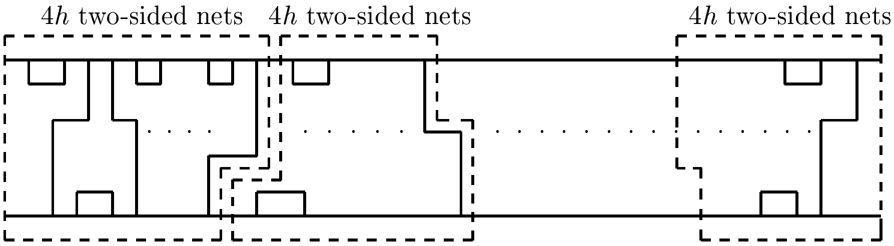

FIG. 5. *The partition for $s < 4h$.*

of two-sided nets are assigned to exactly one of those blocks.) Since the upper side and lower side of a block may not be of the same length, we define c_i to be the sum of the number of columns the upper side spans and the number of columns the lower side spans.

From lemma 2.2, $\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} \mathrm{FOP}(s, L_i \cup T \cup U)$. Since $s < 4h$, there must be fewer than $4h$ two-sided nets through any vertical cut at any feasible offset. Therefore, any offset with vertical cuts through L_i and U_j for $j > i + 1$ or $j < i - 1$ would be an unfeasible offset, because such a cut would be crossed by all the nets in T_{i+1} or T_{i-1}. Thus we can write

$$\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} [\mathrm{FOP}(s, L_i \cup T \cup U_i) \cap \mathrm{FOP}(s, L_i \cup T \cup U_{i+1}) \cap \mathrm{FOP}(s, L_i \cup T \cup U_{i-1})] .$$

Note also that no vertical cuts through both L_i and U_j can cut any two-sided nets outside blocks i through j, so we can rewrite $\mathrm{FOP}(s, N)$ as

$$\bigcap_{i=1}^{t/4h} [\mathrm{FOP}(s, L_i \cup T_i \cup U_i) \cap \mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup U_{i+1}) \cap \mathrm{FOP}(s, L_i \cup T_{i-1} \cup T_i \cup U_{i-1})] .$$

Now we can solve each of $\mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup U_{i+1})$ and $\mathrm{FOP}(s, L_i \cup T_{i-1} \cup T_i \cup U_{i-1})$ in time $O(h(c_{i-1}+c_i+c_{i+1}) \lg(c_{i-1}+c_i+c_{i+1}))$ as follows. We use Lemma 3.4 to further decompose the problem, Lemma 2.1 for the computation of new contours (which will still have $O(h)$ extensions), and Lemmas 3.1 and 2.7 to solve the subproblems.

To solve $\mathrm{FOP}(s, L_i \cup T_i \cup U_i)$, we use a recursive method, for which we consider the general problem of solving $\mathrm{FOP}(s, L^* \cup T^* \cup U^*)$ with $|T^*| = t^* \leq 4h$. We decompose such a problem into left and right blocks, each having half as many two-sided nets as the original. Using subscripts l and r to denote the portions of L^*, T^*, and U^* falling in the left and right blocks, we know from Lemma 2.2 that

$$\begin{aligned}\mathrm{FOP}(s, L^* \cup T^* \cup U^*) &= \mathrm{FOP}(s, L_l^* \cup T^* \cup U^*) \cap \mathrm{FOP}(s, L_r^* \cup T^* \cup U^*)\\ &= \mathrm{FOP}(s, L_l^* \cup T_l^* \cup U_l^*) \cap \mathrm{FOP}(s, L_l^* \cup T^* \cup U_r^*) \cap\\ &\ \mathrm{FOP}(s, L_r^* \cup T^* \cup U_l^*) \cap \mathrm{FOP}(s, L_r^* \cup T_r^* \cup U_r^*)\ .\end{aligned}$$

The restrictions from T^* to T_l^* and T_r^* are by reasoning similar to that used above. Also as above, we use Lemma 3.4, Lemma 2.1, Lemma 3.1, and Lemma 2.7 to solve $\mathrm{FOP}(s, L_l^* \cup T^* \cup U_r^*)$ and $\mathrm{FOP}(s, L_r^* \cup T^* \cup U_l^*)$ in time $O(hc^* \lg c^*)$, where c^* is the sum of the number of top and bottom columns spanned by L^*, T^*, and U^*. $\mathrm{FOP}(s, L_l^* \cup T_l^* \cup U_l^*)$ and $\mathrm{FOP}(s, L_r^* \cup T_r^* \cup U_r^*)$ are just recursive calls; we denote the sum of the number of top and bottom columns in these subproblems as c_l^* and c_r^*. Then, the time $T(t^*, c^*)$ to compute $\mathrm{FOP}(s, L^* \cup T^* \cup U^*)$ can be written as

$$T(t^*, c^*) = T(t^*/2, c_l^*) + T(t^*/2, c_r^*) + O(hc^* \lg c^*)$$

for all $t^* \leq 4h$, where $c_l^* + c_r^* = c^*$, and $T(1, c) = O(hc \lg c)$. There are $O(\lg t^*)$ stages in the recursion, and the total work on all subproblems at any given stage is $O(hc^* \lg c^*)$ time. Thus, $T(t^*, c^*) = O(hc^* \lg c^* \lg t^*)$, and, in particular, $T(4h, c_i) = O(hc_i \lg c_i \lg h) = O(hc_i \lg^2 c)$.

Putting everything together, the time to solve $\mathrm{FOP}(s, N)$ is

$$\begin{aligned}T(t, c) &= \sum_{i=1}^{t/4h} T(4h, c_i) + \sum_{i=1}^{t/4h} O((c_{i-1} + c_i + c_{i+1}) h \lg c)\\ &= \sum_{i=1}^{t/4h} O(hc_i \lg^2 c) + O(hc \lg c)\\ &= O(hc \lg^2 c) + O(hc \lg c)\\ &= O(hc \lg^2 c).\end{aligned}$$

Finally, we must return to solving the original problem in the case that $s \geq 4h$. We divide the channel into $t/2h$ blocks, each spanning $2h$ two-sided nets. From Lemma 2.2, $\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/2h} \mathrm{FOP}(s, L_i \cup T \cup U)$. Furthermore, at any offset, we need not consider any vertical cut for which the number of two-sided nets crossing the cut is less than $s - 2h$ or greater than s. In the former case, we know the cut cannot provide evidence of infeasibility; in the latter case infeasibility is guaranteed. Thus, we can write

$$\begin{aligned}\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/2h} &[\mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup \cdots \cup T_{i+s/2h} \cup U_{i+s/2h-1} \cup U_{i+s/2h}) \cap\\ &\mathrm{FOP}(s, L_i \cup T_i \cup T_{i-1} \cup \cdots \cup T_{i-s/2h} \cup U_{i-s/2h+1} \cup U_{i-s/2h})].\end{aligned}$$

At this point, we could proceed as when $s < 4h$ by incorporating two-sided nets into the upper and lower contours, but there are too many two-sided nets to get the desired running time; we might end up with more than $O(h)$ distinct extensions in the contours. Instead, we take out the $s - 4h$ two-sided nets between L_i and $U_{i+s/2h-1}$, and the $s - 4h$ two-sided nets between L_i and $U_{i-s/2h+1}$, and we decrease s by $s - 4h$.

Each unfeasible offset in this new problem actually denotes the center of a range of $2(s-4h)+1$ unfeasible offsets of the original problem, but with this proviso, the task is to solve

$$\bigcap_{i=1}^{t/2h}[\quad \mathrm{FOP}(4h,\quad L_i \cup T_i \cup T_{i+s/2h-1} \cup T_{i+s/2h} \cup U_{i+s/2h-1} \cup U_{i+s/2h}) \cap$$
$$\mathrm{FOP}(4h,\quad L_i \cup T_i \cup T_{i-s/2h+1} \cup T_{i-s/2h} \cup U_{i-s/2h+1} \cup U_{i-s/2h})]\ .$$

We can solve these subproblems using Lemmas 3.4, 2.1, 3.1, and 2.7 as before. Also, with a similar analysis for the combined running time of the subproblems, we get a total time of $O(hc\lg c)$. ∎

THEOREM 3.6. FOP *can be solved in* $O(c\sqrt{n}\lg c)$ *time.*

Proof. We can use Lemma 2.2 to write $FOP(s,N) = FOP(s, L_l \cup T \cup U_l) \cap FOP(s, L_g \cup T \cup U) \cap FOP(s, U_g \cup T \cup L)$, where L_l is the subset of L with extensions less than h, and L_g is the subset of L with extensions greater than or equal to h, and similarly for U_l and U_g. The first subproblem can be solved in $O(hc\lg^2 c)$ time using Lemma 3.5, and the latter two subproblems can be solved in $O(cn/h)$ time using Lemma 3.3. By letting $h = \sqrt{n}/\lg c$, FOP can be solved in $O(c\sqrt{n}\lg c)$ time. ∎

We now show how to use a halving technique similar to that of [11] and [9] to solve OOP in the same time as FOP. We actually focus here on finding optsep(P), the minimum separation attainable with an optimal offset for the routing problem P; once optsep(P) is determined, the solution of the feasible offset problem can be used to determine the optimal offsets. From the original problem P, we create a simpler problem P^e that has about half the separation of P. The basic idea is to halve the extensions of the contours of single-sided nets, remove every other two sided net, and compact the channel horizontally to eliminate the freed space. More precisely, if the two sided nets are numbered 1 through t from left to right, we remove all the odd-numbered nets and move the terminals of net $2i$ to the left by i units. The nonzero portions of the single-sided contours are also shifted left so that they stay the same distance from their nearest two-sided nets. This is the same transformation as used in [9], but the effect on optsep is slightly different here due to the general arrangement of single-sided nets and the disallowance of routing on the channel boundaries, and the timing analysis for computing optsep(P) differs more substantially. The following lemma states the relationship between optsep(P) and optsep(P^e):

LEMMA 3.7. *Let* $s = \mathrm{optsep}(P)$ *and* $s^e = \mathrm{optsep}(P^e)$. *Then* $2s^e - 2 \le s \le 2s^e + 2$.

Proof. We again use the theory of single-layer routability from [10]. In our context, the *flow* of a cut is the number of nets that *must* cross it, and the cut is *safe* if its flow is no greater than one less than the maximum of the horizontal and vertical extents of the cut. A cut χ in P that crosses f nets, p of which are lower nets, q of which are two-sided nets, and r of which are upper nets can be seen to correspond to a cut χ^e with the following properties: (1) The flow of χ^e is in the range $[\frac{p-1}{2}+\frac{q-1}{2}+\frac{r-1}{2}, \frac{p}{2}+\frac{q+1}{2}+\frac{r}{2}] = [\frac{f}{2}-\frac{3}{2}, \frac{f}{2}+\frac{1}{2}]$, and (2) the horizontal extent of χ^e is diminished relative to χ to the same extent as the flow. Thus $s^e - 1 \in [\frac{s-1}{2}-\frac{3}{2}, \frac{s-1}{2}+\frac{1}{2}]$, i.e., $2s^e \in [s-2, s+2]$. ∎

THEOREM 3.8. OOP *can be solved in* $O(c\sqrt{n}\lg c)$ *time.*

Proof. To find optsep(P), we recursively find $s^e = \mathrm{optsep}(P^e)$. Then we need only determine which of the five separations in $[2s^e-2, 2s^e+2]$ have feasible offsets for P. Let $T(m)$ be the solution time for optsep(P) where P has maximum extension m but has been derived by repeated application of the halving transformation to a problem with maximum extension M. Then any extension h in P has been derived from an extension hM/m in the original problem. An argument as in the Proof of Theorem 3.6 then tells us that we can solve FOP for P in $O(\sqrt{m}\sqrt{n/M}c\lg c)$ time. Thus we have $T(m) = T(m/2) + \sqrt{m}\sqrt{n/M}c\lg c$, which yields $T(m) = \sqrt{m}\sqrt{n/M}c\lg c \leq \sqrt{n}c\lg c$. ∎

COROLLARY 3.9. FOP *and* OOP *can be solved in* $O(n^{1.5}\lg n)$ *time for* $c = O(n)$. ∎

4. Conclusion and Further Results. We have shown how to solve the feasible offset and optimal offset problems for single-layer channel routing in time $O(c\sqrt{n}\lg c)$. (The time is $O(c\sqrt{n\lg c})$ when all nets are single-sided and $O(n)$ when all single-sided nets are on one side of the channel.) This result is unattractive for large values of c, but there is a superior alternative when c is larger than $n^{1.5}$. Essentially all the necessary machinery is already in place for the following result that FOP can be solved in $O(n^2\lg n)$ time independent of the number of columns:

THEOREM 4.1. FOP *can be solved in* $O(n^2\lg n)$ *time.*

Proof. Using Lemma 2.2, we decompose the lower contour into maximal monotonic subsets. Since there are only n nets, we have at most n monotonic subsets. We can find the feasible offsets for each subset in $O(n)$ time using Lemma 3.2. The total time to find the feasible offsets for all of the subsets of the lower contour is $O(n^2)$. Furthermore, for each subset, the set of feasible offsets can be output as a list of at most n nonoverlapping intervals with all the interval endpoints in sorted order. Two sets of nonoverlapping intervals with endpoints in sorted order can be intersected in time proportional to the total number of intervals, which is an upper bound on the output size. We intersect the $O(n)$ sets of intervals in a tournament style, i.e., we go from n sets with n intervals in each set, to $n/2$ sets with $2n$ intervals in each sets,. . ., to 1 set with n^2 intervals. There are $\lg n$ stages, with $O(n^2)$ work at each stage, yielding a total time of $O(n^2\lg n)$. ∎

One direction for further research is to improve the time for feasible offset when the number of columns is large. We know that $\Omega(n^2)$ is a lower bound on the worst case running time, but we suspect that it may not be difficult to obtain an $O(n^2)$ upper bound as well. Another remaining open question is whether our upper bounds for feasible offset with smaller c can be improved. We know of no nontrivial lower bounds, i.e., better than $\Omega(\min\{c, n^2\})$. It also might be possible to improve the time to solve optimal offset without making further progress on feasible offset. Though it seems unlikely that optimal offset would be much easier than feasible offset, optimal offset has a much smaller output size, and output size is the only basis for our lower bounds on feasible offset.

It is also desirable to handle the situation in which there are multiple modules. Within each module, the positions of the terminals are fixed, but, on each side of the channel, the modules can slide back and forth as long as their order does not change. In the *optimal placement problem*, the goal is to minimize the channel length given a channel width. We can solve this problem in $O(n^3)$ time by adapting ideas used by Chao and LaPaugh [3] for density minimization; more details can be found in [13]. A further direction for research is to improve this $O(n^3)$ time when there is a reasonable bound on the number of columns.

Finally, an interesting open problem related to SSFOP is efficient computation of the max, + convolution. The technique in Lemma 2.7 can be extended to yield a solution to max, + convolution for n-vectors of small integers (e.g., $\leq n^{1/4}$) in less than $\Theta(n^2)$ time. If the range of integers could be extended to 1 through n, improved solutions for several VLSI routing problems would result (e.g., see [11]).

REFERENCES

[1] B. S. Baker and R. Y. Pinter, *An algorithm for the optimal placement and routing of a circuit within a ring of pads*, in 24th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1983, pp. 360–370.

[2] S.-C. Chang, J. JáJá, and K. W. Ryu, *Optimal parallel algorithms for one-layer routing*, Tech. Report UMIACS-TR-89-46, University of Maryland Institute for Advanced Computer Studies, Apr. 1989.

[3] L.-F. Chao and A. S. LaPaugh, *Finding all minimal shapes in a routing channel*, Tech. Report CS-TR-384-92, Princeton University Department of Computer Science, Aug. 1992.

[4] R. Cole and A. Siegel, *River routing every which way, but loose*, in 25th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1984, pp. 65–73.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1990.

[6] D. Dolev, K. Karplus, A. Siegel, A. Strong, and J. D. Ullman, *Optimal algorithms for structural assembly*, VLSI Design, (1982), pp. 38–43. Earlier version in *Proceedings of the 13th ACM Symposium on Theory of Computing*.

[7] R. I. Greenberg, A. T. Ishii, and A. L. Sangiovanni-Vincentelli, *MulCh: A multi-layer channel router using one, two, and three layer partitions*, in IEEE International Conference on Computer-Aided Design (ICCAD-88), IEEE Computer Society Press, 1988, pp. 88–91.

[8] R. I. Greenberg and F. M. Maley, *Minimum separation for single-layer channel routing*, Information Processing Letters, 43 (1992), pp. 201–205.

[9] R. I. Greenberg and J.-D. Shih, *Single-layer channel routing and placement with single-sided nets on one side*, Tech. Report UMIACS-TR-93-6, University of Maryland Institute for Advanced Computer Studies, Jan. 1993. Revised version submitted.

[10] F. M. Maley, *Single-Layer Wire Routing and Compaction*, MIT Press, 1990.

[11] A. Mirzaian, *River routing in VLSI*, Journal of Computer and System Sciences, 34 (1987), pp. 43–54.

[12] R. Y. Pinter, *The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits*, PhD thesis, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, Aug. 1982. MIT/LCS/TR-291.

[13] J.-D. Shih, *Efficient Algorithms for Channel Routing and Placement Problems*, PhD thesis, University of Maryland Electrical Engineering Department, 1993.