6-1993

Feasible Offset and Optimal Offset for Single-Layer Channel Routing

Ronald I. Greenberg
Loyola University Chicago, Rgreen@luc.edu

Jau-Der Shih

Recommended Citation

Greenberg, Ronald I. and Shih, Jau-Der. Feasible Offset and Optimal Offset for Single-Layer Channel Routing. Proceedings of the 2nd Annual Israel Symposium on Theory of Computing and Systems, , : 193-201, 1993. Retrieved from Loyola eCommons, Computer Science: Faculty Publications and Other Works, http://dx.doi.org/10.1109/ISTCS.1993.253470

Feasible Offset and Optimal Offset for Single-Layer Channel Routing

Ronald I. Greenberg and Jau-Der Shih
Electrical Engineering Department
University of Maryland
College Park, MD 20742

Prepub manuscript for Proceedings of 2nd Annual Israel Symposium on Theory of Computing and Systems, 1993, pp. 193–201

Abstract

This paper provides an efficient method to find all feasible offsets for a given separation in a single-layer channel routing problem. With n nets, c columns, and $c = O(n)$, the algorithm runs in time $O(n^{1.5} \lg n)$, which improves upon the more straightforward $O(n^2)$ approach. (Results for river routing do not apply, since we allow single-sided nets.) As a corollary of this result, the optimal offset (the one that minimizes separation) can be found in $O(n^{1.5} \lg^2 n)$ time. We also provide results for the case in which $c \neq O(n)$. In addition, we achieve better running times for the cases in which there are no two-sided nets or all single-sided nets are on one side of the channel.

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., [12, 14, 11, 9, 5, 15]. 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, 8, 3].

Ironically, single-layer routing may become more relevant as technology evolves towards chips with increasing numbers of layers. With a larger number of 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 [6] 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. But before exploring this distinction, it is important to recognize that "routing problem" is used in a loose sense in this paper and throughout the literature. That is, given fixed (and feasible) positions of the terminals, it is relatively easy to find actual paths for all the requisite connecting wires in the optimal time of $O(n^2)$. (Here n is the number of nets, or if nets can have more than a constant number of terminals, n should represent the number of terminals.) But more interesting versions of the problem involve some form of

[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.

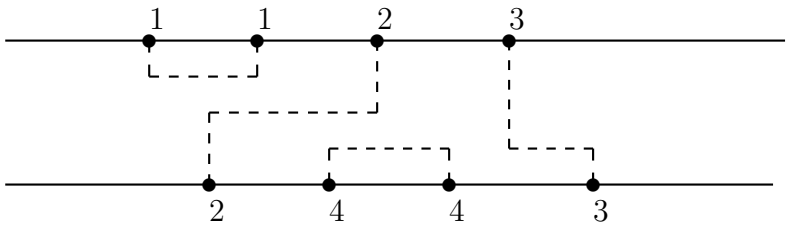

Figure 1: An example of single-layer channel routing

optimization of the terminal positions subject to certain constraints and generally run faster than just $O(n^2)$.

The problem inputs for channel routing and river routing are similar 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. 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 [6]) and a significant algorithmic complication. Figure 1 shows an example of a routed single-layer channel. As the figure shows, the usual convention is to draw the rows of terminals horizontally, so that the terminals lie on the top and bottom sides of the channel. We refer to single-sided nets that have their two terminals on the top (bottom) as *upper (lower) nets*. Nets with terminals on opposite sides, the only type of net in river routing problems, are referred to as two-sided nets. Throughout this paper 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 separation between the two rows of terminals (also referred to as *channel width*) that allows the routing to be completed. An $O(n)$ time solution in the river routing case was given in [5]. The attempt to extend this result to channel routing has lead to publication of erroneous solutions in the literature; but a simple and correct $O(n)$ algorithm is provided in [7].

In this paper, we extend two other important versions of the river routing problem to the channel routing case. In both of 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 both 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 of the complications 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 in the case of feasible offset). As illustrated in Figure 2, if the number of columns can be large, the number of feasible offsets may be on the order of n^2. But if $c = O(n)$, we show that feasible offset can be

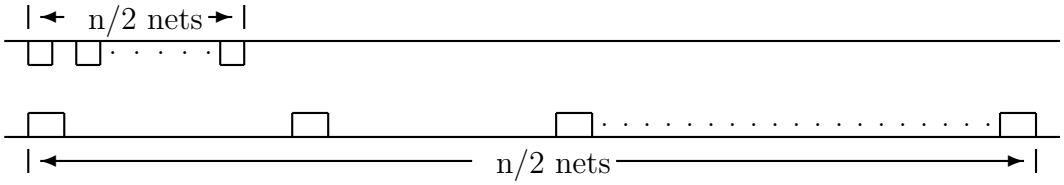

Figure 2: The number of feasible offsets of the channel above is on the order of n^2

solved in $O(n^{1.5} \lg n)$ time. This is an improvement upon the relatively naive $O(cn)$ time ($O(n^2)$ when $c = O(n)$) 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 $c = O(n)$. For somewhat larger c, the results of Sections 2 and 3 remain attractive, but we show in Section 4 that when c is very large, we are better off discarding some of this work in favor of an approach that solves the problem in $O(n^2 \lg n)$ time.

The remainder of this paper is organized as follows. In Section 2, we introduce 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 $O((n \lg n)^{1.5})$ algorithm for optimal offset. 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 times for feasible offset and optimal offset become $O(n^{1.5} \lg n)$ and $O(n^{1.5} \lg^2 n)$, respectively. Finally, in Section 4, we provide concluding remarks, including discussion of the situation in which c is much larger than $O(n)$.

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 these two sections.

First, we often need to refer to subsets of the nets in the channel. 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. (Reminders of this notation appear elsewhere in the paper to keep different portions more self-contained.) 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 these 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 in the case that all nets are single-sided.

Our first step in solving SSFOP is to find the contours of the upper and lower nets. The lemma below indicates that we can find contours in $O(n)$ time, by which we mean finding the coordinates

of all the bends in the contour. (In the sequel, we will sometimes assume that we have generated the entire path of the contour across all c columns, which is a trivial $O(c)$ time task once we have found all the bends.)

Lemma 1 (Pinter) *The contour of a set of n single-sided nets can be found in $O(n)$ time.*

Proof. This is proved in [13]. ■

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 convolution of the two sequences of extensions (after padding them out with some 0's) with the max and $+$ operators substituted for the usual $+$ and $\times$. But it is unknown whether $max, +$ convolution can be computed in less than $O(n^2)$ time. We'll solve SSFOP more efficiently by making use of the fact that the extensions are not completely arbitrary, but rather, come from a set of single-sided nets.

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 *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. If an offset is not feasible for any subset, then it is not feasible for the whole set. Therefore, $\mathrm{FOP}(s, L \cup A) \subseteq \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup A)$. Conversely, if an offset is not feasible for the whole set, we can find the column where the sum of the upper and lower extensions is greater than s and the subset containing that lower extension. The offset will be unfeasible for that subset, thus $\mathrm{FOP}(s, L \cup A) \supseteq \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup A)$. ■

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 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 sketch. Without loss of generality, assume the monotonic piece has (nondecreasing) extensions $l_a, l_{a+1}, \ldots, l_{a+k}$ from left to right, where subscripts indicate the columns covered, and denote the upper extensions by $u_1, u_2, \ldots, u_c$. For simplicity, we pad out the sequence of upper extensions with c 0's on both sides and denote the new sequence as $u_{-c+1}u_{-c+2}\cdots u_{2c}$. Then the procedure in Figure 3 solves the problem in $O(c)$ time. The idea of the procedure is to consider vertical

```
1    q ← 0
2    for i ← −c to 2c do
3        while q > 0 and l_{a+q} + u_i > s
4            q ← q − 1
5        endwhile
6        if l_{a+q} + u_i ≤ s
7        then
8            q ← q + 1
9        endif
10       if q = k + 1
11       then
12           mark i − (a + k) as a feasible offset
13           q ← q − 1
14       endif
15   endfor
```

Figure 3: The pseudocode to solve SSFOP for a monotonic subset of L.

cuts that intersect the monotonic piece of contour as it slides across the channel from a far left position (highly negative offset) to a far right position. As the monotonic lower contour moves left to right, we check the relevant extensions of the upper contour in a left to right fashion as well. We never have to recheck an upper extension that did not lead to infeasibility at a previous offset, since the lower contour is nondecreasing. Furthermore, each time that an upper extension does lead to infeasibility, we know that we should increment the offset, and we can only increment the offset $O(c)$ times. (Offsets beyond c columns in either direction are guaranteed to be feasible.) Though we have only shown a running time of $O(c)$, we can actually solve the problem in $O(n)$ time, because we really only need to look at columns where the upper contour bends. ■

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 with extensions greater than or equal to h.

Lemma 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 extension 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. There are totally c columns, so L_g contains at most $c/2h$ maximal monotonic pieces. ■

We now put together Lemmas 3 and 4 to show that we can solve SSFOP efficiently for any piece of lower contour in which all extensions are large enough.

Lemma 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* SSFOP$(s, L_g \cup U)$ *in $O(cn/h)$ time.*

Proof. By Lemma 2, we know that it suffices to solve the problem independently for each of the maximal monotonic pieces of L_g. By Lemma 4 there are $O(c/h)$ such pieces, and by Lemma 3, $O(n)$ time suffices for each piece. ■

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 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 as explained in [4], for example. ■

Lemma 7 *If all extensions of the lower contour are less than or equal to* h*, then we can solve* SSFOP *in* $O(hc \lg c)$ *time.*

Proof. From lemma 2, $\text{SSFOP}(s, L \cup U) = \bigcap_{i=1}^{h} \text{SSFOP}(s, L_i \cup U)$, where L_i is the subset of the lower contour with extension i. We can now solve $\text{SSFOP}(s, L_i \cup U)$ using lemma 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 $O(hc \lg c)$. ■

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

Theorem 8 SSFOP *can be solved in* $O(c\sqrt{n \lg c})$ *time.*

Proof. $\text{SSFOP}(s, L \cup U) = \text{SSFOP}(s, L_l \cup U) \cap \text{SSFOP}(s, L_g \cup U)$, 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. $\text{SSFOP}(s, L_l \cup U)$ can be solved in $O(hc \lg c)$ time using lemma 7, and $\text{SSFOP}(s, L_g \cup U)$ can be solved in $O(cn/h)$ time using lemma 5. By letting $h = \sqrt{n/\lg c}$, the combined time is $O(c\sqrt{n \lg c})$. ■

Corollary 9 SSFOP *can be solved in* $O(n^{1.5}\sqrt{\lg n})$ *time for* $c = O(n)$. ■

Corollary 10 SSOOP *can be solved in* $O((n \lg n)^{1.5})$ *time when* $c = O(n)$.

Proof. The optimal offset problem can be solved by performing binary search on separation and seeing whether there are any feasible offsets at each stage. Since the separation cannot exceed n, the run time is $O(\lg n)$ times the time for SSFOP. ■

3 General Single-Layer Channel

In this section, we use the ideas of section 2 to solve FOP and OOP in the more general case in which there are two-sided as well as single-sided nets. As before, we begin by computing the contours of the upper nets and the lower nets. As in the single-sided case, we consider separately the portions of the lower contour with large extensions and the portions with small extensions and then show how to put the two 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.

(a) (b)

Figure 4: The effect of two-sided nets in (a) is incorporated into the top contour in (b).

Lemma 11 *When all single-sided nets are on one side,* FOP *and* OOP *can be solved in* $O(n \lg n)$ *time.*

Proof sketch. The idea is to use binary search on offset as in [5]. At any given offset, define a left-going (right-going) two-sided net as a net whose upper terminal is to the left (right) of the lower terminal. The separation with right-going (left-going) nets omitted can't decrease as the set of upper terminals shifts to the left (right). Consequently, by seeing which of these two separations is at least as large as the other, we can eliminate either all larger or all smaller offsets, while retaining an optimal offset among those offsets still under consideration. It takes $O(n)$ time to find the separation [7], thus $O(n \lg c)$ time suffices to solve OOP, since there are only $2c$ meaningful offsets. Furthermore, as in the case when there are no single-sided nets, there are actually just $O(n^2)$ critical offsets that must be checked, so the time can be reduced to $O(n \lg n)$ as in [5]. FOP can be solved similarly. ■

In order 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 all to the left of L_0. Define T/L_0 to be the set of nets obtained by pulling up the lower terminals of the nets in T and reconnecting them to the upper side to the left of any 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 T of two-sided nets and any piece C of either the upper or lower contour, whether T is to the left or right of C. In all cases T/C is a set of single-sided nets formed by moving terminals in T on the same side as C away from C. Figure 4 illustrates the definition of T/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 with extension s.

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 12 *Let A be a set of upper and two-sided nets. Then after $O(n \lg n)$ preprocessing, we can solve* 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: (1) Solve FOP without L_m. (2) Solve FOP by considering (at each offset) only those columns L_m intersects. Subproblem (1) is solved by Lemma 11 with $O(n \lg n)$ preprocessing. The idea to solve subproblem (2) is to reroute the two-sided nets in the fashion shown in Figure 4. By doing so we incorporate into the upper contour any effect of two-sided nets that can come into play with L_m. But we may be building up the upper contour excessively in columns beyond the range of L_m, which effect is

eliminated by truncating the new upper contour to extension s. (This truncation does not hurt when columns of the upper contour fall into the range of L_m, since any upper extension of s or greater will be enough to flag infeasibility when matched to any (nonzero) extension of L_m.) By this reasoning, the solution to subproblem (2) is given by $\mathrm{SSFOP}(s, L_m \cup (U \cup T/L_m)|_s)$. Using lemma 1 and lemma 3, we can solve $\mathrm{SSFOP}(s, L_m \cup (U \cup T/L_m)|_s)$ in $O(n)$ time. ■

Now we can solve FOP for a subset L with large extensions. As before, we define large as exceeding h and specify the value of h later.

Lemma 13 *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 $O(cn/h + n \lg n)$ time.*

Proof. By lemma 2, L_0 can be decomposed into maximal monotonic subsets $L_1, L_2, \ldots, L_k$. Using lemma 12, we have

$$\begin{aligned}
\mathrm{FOP}(s, L_g \cup U \cup T) &= \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup U \cup T) \\
&= \bigcap_{i=1}^{k} [\mathrm{FOP}(s, U \cup T) \cap \mathrm{FOP}(s, L_i \cup (U \cup T/L_i)|_s)] \\
&= \mathrm{FOP}(s, U \cup T) \cap \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup (U \cup T/L_i)|_s) \ .
\end{aligned}$$

Now, k is $O(c/h)$ by lemma 4. $\mathrm{FOP}(s, U \cup T)$ corresponds to the $O(n \lg n)$ preprocessing in Lemma 12, and each $\mathrm{FOP}(s, L_i \cup (U \cup T/L_i)|_s)$ can be solved in $O(n)$ time by Lemma 12. Thus, the total running time is $O(cn/h + n \lg n)$. ■

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 14 *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. Then,*

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

Proof. The argument is similar to the one for lemma 12. 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., we solve $\mathrm{FOP}(s, (U_l \cup T_l/L_r)|_s \cup (L_r \cup T_r/U_l)|_s)$ as illustrated in Figure 5. 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.

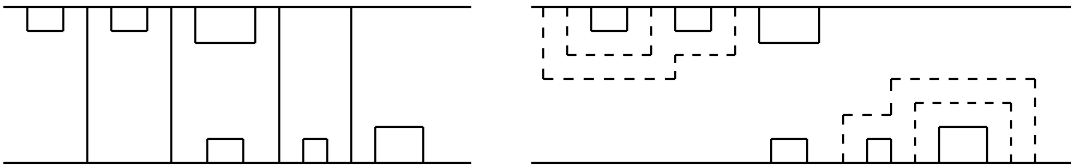

Figure 5: The effect of two-sided nets in (a) is incorporated into the upper and lower contours in (b).

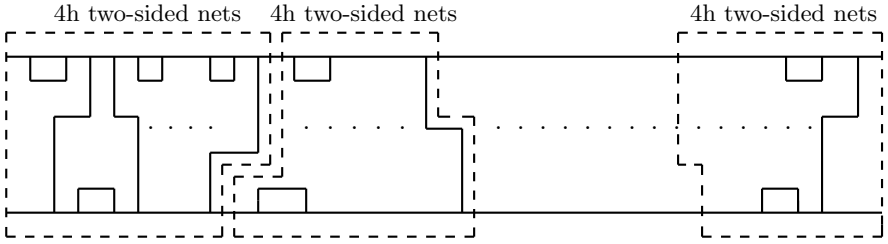

Figure 6: The partition for $s < 4h$.

Lemma 15 *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 6. Let L_i, U_i, and T_i denote the lower single-sided nets, upper single-sided nets, and two-sided nets in block i. (Single-sided nets at a boundary between blocks 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, $\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} \mathrm{FOP}(s, L_i \cup T \cup U)$. 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 14 to further decompose the problem, Lemma 1 for the computation of new contours (which will still have $O(h)$ extensions), and Lemmas 11 and 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 that

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

The restrictions from T to T_l^* and T_r^* are by reasoning similar to that used above. Also as above, we use Lemma 14, Lemma 1, Lemma 11, and Lemma 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 to compute $\mathrm{FOP}(s, L^* \cup T^* \cup U^*)$, $T(t^*, c^*)$, can be written as

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

where $c_l^* + c_r^* = c^*$, and $T(1, c) = O(hc \lg c)$. There are $O(\lg h)$ 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 $s/2h$ blocks, each spanning $2h$ two-sided nets. From Lemma 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

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

We can solve these subproblems using Lemmas 14, 1, 11, and 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 16 FOP *can be solved in* $O(c\sqrt{n} \lg c)$ *time.*

Proof. We can use Lemma 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 15, and the latter two subproblems can be solved in $O(cn/h + n \lg n)$ time using Lemma 13. By letting $h = \sqrt{n}/\lg c$, FOP can be solved in $O(c\sqrt{n} \lg c)$ time. ■

Corollary 17 FOP *can be solved in* $O(n^{1.5} \lg n)$ *time for* $c = O(n)$. ■

Corollary 18 OOP *can be solved in* $O(n^{1.5} \lg^2 n)$ *time when* $c = O(n)$.

Proof. The optimal offset problem can be solved by binary search on separation, thus the time bound is $O(n^{1.5} \lg^2 n)$. ■

4 Conclusion and Further Results

We have shown how to solve the feasible offset and optimal offset problems for single-layer channel routing with $c = O(n)$ in time $O(n^{1.5} \lg n)$ and $O(n^{1.5} \lg^2 n)$, respectively. (We have also obtained times of $O(n^{1.5}\sqrt{\lg n})$ and $O((n \lg n)^{1.5})$ for the case in which all nets are single-sided and $O(n \lg n)$ for both problems when all single-sided nets are on one side of the channel.) Furthermore, we have shown how to solve the feasible offset problem in $O(c\sqrt{n} \lg c)$ time for any value of c (and $O(c\sqrt{n} \lg c \lg n)$ time for optimal offset), which improves upon the more naive bound of $O(cn)$ except for truly huge values of c. But, we can do even better than $O(c\sqrt{n} \lg c)$ for moderately large values of c, i.e., larger than about $n^{1.5}$. Essentially all the necessary machinery is already in place for this further result that FOP can be solved in $O(n^2 \lg n)$ time independent of the number of columns.

Theorem 19 FOP *can be solved in* $O(n^2 \lg n)$ *time.*

Proof. Using lemma 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 12 after $O(n \lg n)$ preprocessing time. The total time to find the feasible offsets for all of the subsets of the lower contour is $O(n^2)$. Now, each subset has a set of feasible offsets that can be expressed as at most $2n$ nonoverlapping intervals with the endpoints of the intervals 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 $2n$ intervals in each set, to $n/2$ sets with $4n$ 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. Finally, it might be possible to improve our running time for problems with all the single-sided nets on one side. Though $O(n \lg n)$ time was a sufficient tool for solving the general problem, it may be possible to bring the time down to $O(n)$ for problems with all single-sided nets on one side.

References

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

[2] Shing-Chong Chang, Joseph JáJá, and Kwan Woo Ryu. Optimal parallel algorithms for one-layer routing. Technical Report UMIACS-TR-89-46, University of Maryland Institute for Advanced Computer Studies, April 1989.

[3] Richard Cole and Alan Siegel. River routing every which way, but loose. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, pages 65–73. IEEE Computer Society Press, 1984.

[4] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1990.

[5] Danny Dolev, Kevin Karplus, Alan Siegel, Alex Strong, and Jeffrey D. Ullman. Optimal wiring between rectangles. In *Proceedings of the 13th ACM Symposium on Theory of Computing*, pages 312–317. ACM Press, 1981.

[6] Ronald I. Greenberg, Alexander T. Ishii, and Alberto L. Sangiovanni-Vincentelli. MulCh: A multi-layer channel router using one, two, and three layer partitions. In *Proceedings of the IEEE International Conference on Computer-Aided Design (ICCAD-88)*, pages 88–91. IEEE Computer Society Press, 1988.

[7] Ronald I. Greenberg and F. Miller Maley. Minimum separation for single-layer channel routing. *Information Processing Letters*, 43(4):201–205, September 1992.

[8] Charles E. Leiserson and F. Miller Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 69–78. ACM Press, 1985.

[9] Charles E. Leiserson and Ron Y. Pinter. Optimal placement for river routing. *SIAM Journal on Computing*, 12(3):447–462, August 1983.

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

[11] Andranik Mirzaian. River routing in VLSI. *Journal of Computer and System Sciences*, 34:43–54, 1987.

[12] Andranik Mirzaian. A minimum separation algorithm for river routing with bounded number of jogs. In *Proceedings of the IEEE International Conference on Computer-Aided Design (ICCAD-89)*, pages 10–13. IEEE Computer Society Press, 1989.

[13] Ron Yair 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, August 1982. MIT/LCS/TR-291.

[14] Alan Siegel and Danny Dolev. Some geometry for general river routing. *SIAM Journal on Computing*, 17(3):583–605, June 1988.

[15] Martin Tompa. An optimal solution to a wire-routing problem. *Journal of Computer and System Sciences*, 23:127–150, 1981.