PARALLEL ALGORITHMS FOR SINGLE-LAYER CHANNEL ROUTING

RONALD I. GREENBERG
Dept. of Mathematical and Computer Sciences, Loyola University
6525 N. Sheridan Rd., Chicago, IL 60626, USA
E-mail: rig@math.luc.edu

and

SHIH-CHUAN HUNG
Quickturn Design Systems, Inc.
440 Clyde Ave., Mountain View, CA 94041, USA
E-mail: shih@quickturn.com

and

JAU-DER SHIH
Dept. of Mathematics and Science Education, National Pingtung Teachers College,
No. 1 Lin Sen Road, Pingtung, Taiwan, Republic of China
E-mail: jdshih@sparc5.npttc.edu.tw

ABSTRACT

We provide efficient parallel algorithms for the minimum separation, offset range, and optimal offset problems for single-layer channel routing. We consider all the variations of these problems that are known to have linear-time sequential solutions rather than limiting attention to the "river-routing" context, where single-sided connections are disallowed. For the minimum separation problem, we obtain $O(\lg N)$ time on a CREW PRAM or $O(\frac{\lg N}{\lg \lg N})$ time on a (common) CRCW PRAM, both with optimal work (processor-time product) of $O(N)$, where N is the number of terminals. For the offset range problem, we obtain the same time and processor bounds as long as only one side of the channel contains single-sided nets. For the optimal offset problem with single-sided nets on one side of the channel, we obtain time $O(\lg N \lg \lg N)$ on a CREW PRAM or $O(\frac{\lg N}{\lg \lg N})$ time on a CRCW PRAM with $O(N \lg \lg N)$ work. Not only does this improve on previous results for river routing, but we can obtain an even better time of $O((\lg \lg N)^2)$ on the CRCW PRAM in the river routing context. In addition, wherever our results allow a channel boundary to contain single-sided nets, the results also apply when that boundary is ragged and N incorporates the number of bendpoints.

Keywords: Parallel algorithms, single-layer channel routing, VLSI layout

1 Introduction

Much attention has been given to single-layer routing for VLSI. Most popular has been river routing [9], the connection of two (horizontal) rows of corresponding terminals using the channel region between the rows of terminals; see also [17] and the references therein. More general arrangements of modules and nets have been considered for testing routability of terminals in fixed positions, but it is also desirable to answer more sophisticated questions. For example, the *minimum separation* problem involves finding the minimum vertical separation between two rows of terminals that is required for routability (given that the horizontal positions of the terminals are completely fixed). In other problems, we are allowed to offset the upper row of terminals as a block to the left or the right, though the individual terminals do not shift position relative to one another. In particular, the *optimal offset problem* involves finding the offset that minimizes the amount of separation necessary to route the channel. The *offset range problem* involves finding all offsets that give enough room to route at a given separation, referred to as the *feasible offsets*.

We consider these problems in all contexts for which linear-time sequential algorithms are known instead of considering only river routing, where each net is restricted to have exactly one terminal on each side of the channel. The input we assume is two arrays of terminals sorted by x-coordinate. The top terminals (and, in arithmetic contexts, their x-coordinates) are denoted $t_1, t_2, \ldots, t_m$, and the bottom terminals are denoted $b_1, b_2, \ldots, b_n$. We use N to denote $m + n$. Associated with each terminal is a net number. Terminals belonging to the same net are to be connected together. We also assume that each terminal has a pointer to the next terminal of the same net in a clockwise ordering of the terminals. Such an assumption is implicit in [7], albeit for two-terminal nets.[a] For simplicity, we use a rectilinear, grid-based model in which terminals lie on gridpoints and wires are disjoint paths through grid edges. Also, for convenience, we allow routing on channel boundaries.

We henceforth assume that each net has two terminals. Multiterminal nets can be handled by a transformation described in [10] (and known previously). Then a *single-sided net* has its two terminals on the same side of the channel, whereas a *two-sided net* is the type of net allowed in river routing.

We also assume henceforth that the channel is routable in one layer, i.e., no two nets are topologically forced to cross. (This condition can be verified without increasing the running time of our parallel algorithm by viewing the first and last terminals of each net as a left parenthesis and right parenthesis, respectively, and testing for proper nesting. The test for proper nesting requires only a prefix sum (discussed further in Section 2) of -1 and $+1$ values for left and right parentheses, respectively.)

The results obtained in this paper and the best corresponding prior results are summarized in Table 1. In that table, "river" refers to the river routing model described above, the "general" model includes any single-layer channel routing problem, and the "intermediate" model is one in which all single-sided nets are on one

[a] It may be possible to eliminate this assumption, with little or no weakening of the results reported in this paper. If the number of terminals per net is bounded by a constant, a simple $O(1)$ time and $O(N)$ work algorithm on the arbitrary CRCW PRAM allows every terminal to find all other terminals of the same net. A fast, optimal, randomized algorithm on the common CRCW PRAM may also follow from the analysis by Berkman et al. [4] of the "c − PARTNERS" problem. Even when the number of terminals per net is not bounded, it may be possible to apply results on sorting of small integers (e.g, [2, 3, 18, 19] and references therein), assuming that nets are numbered using sequential integers rather than arbitrary labels.

Table 1: Running time and work (processor-time product) for the algorithms presented in this paper and related prior work. Note that "N", for example stands for "$O(N)$".

problem	model	prior source	CREW time	CREW work	CRCW time	CRCW work
min. sep.	general	—	$\lg N$	N	$\lg N/\lg\lg N$	N
min. sep.	river	[1, 7]	$\lg N$	N	$\lg\lg N$	N
offset range	intermediate	—	$\lg N$	N	$\lg N/\lg\lg N$	N
offset range	river	[1]	$\lg N$	N	$\lg\lg N$	N
opt. offset	intermediate	—	$\lg N \lg\lg N$	$N\lg\lg N$	$\lg N/\lg\lg N$	$N\lg\lg N$
opt. offset	river	—	$\lg N \lg\lg N$	$N\lg\lg N$	$(\lg\lg N)^2$	$N\lg\lg N$
opt. offset	river	[1]	$\lg^2 N$	$N\lg N$	$\lg N\lg\lg N$	$N\lg N$

side of the channel. Henceforth, CRCW always refers to the *common* CRCW, in which all concurrent writes must be of the same value.

Most prior work on single-layer routing has been limited to sequential models of computation; linear-time sequential algorithms for the problems considered in this paper can be found in [10], [12], and [17]. For the river routing model only, parallel results for the problems in this paper are given by Aggarwal and Park [1]. In addition to obtaining results for more general routing models, this paper improves the time and work bounds for optimal offset. This paper improves those bounds even further when attention is restricted to the river routing model on the CRCW. Chang, JáJá, and Ryu [7], independently obtain the same CREW bounds as Aggarwal and Park for minimum separation in the river routing model and also give an optimal (work) algorithm for routability testing for switchboxes.

The remainder of this paper is organized as follows. In Section 2, we explain the parallel operations used in this paper. We also indicate how to conveniently express the routability conditions for *single-layer channel routing.* These conditions are then used in Section 3 to solve the minimum separation, offset range and optimal offset problems. Section 4 gives some concluding remarks, including a brief explanation of why our results can be readily extended to cases in which channel boundaries containing single-sided nets are ragged (i.e., arbitrary, horizontally monotone, rectilinear boundaries, rather than straight lines.)

2 Preliminaries

2.1 Basic Parallel Operations

Given a sequence of N elements $\{x_1, x_2, \ldots, x_N\}$ with a binary associative operator $*$, the *prefix sums* are all the partial sums defined by:

$$p_i = x_1 * x_2 * \ldots * x_i \qquad 1 \leq i \leq N$$

For a given array $A(i)$ with $1 \leq i \leq N$, the *range maxima problem* is to find the element with maximum value between two given positions i and j. A query can be answered in $O(1)$ time after the preprocessing described in [13].

The range maxima preprocessing can be implemented in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg\lg N})$) time with $O(N)$ work on a CREW (CRCW) PRAM [6]. The prefix sums computation can be performed with the same time and processor bounds on the

CREW [15], and on the CRCW as long as the input elements are integers in the interval $[1, N]$ [8].

Another useful operation is that of finding all nearest smaller values (ANSV). The form of the problem that we will use is as follows. Given a sequence of values $a_1, a_2, \ldots, a_N$, find for each i, the least $j > i$ such that $a_j < a_i$. This problem can be solved with linear work in $O(\lg \lg N)$ time on the CRCW and $O(\lg N)$ time on the CREW [5].

Note also that finding the maximum of N numbers can be done with the same resource bounds as ANSV [13].

2.2 *Cut Conditions*

We need a few definitions in order to use a general theory of single-layer routing developed by Maley [16]. Define a *critical cut* to be a line segment that connects a top and bottom terminal or runs from a terminal straight across to the opposite side of the channel. Define a *pivotal cut* to be a line segment that connects a top and bottom terminal or runs at 45° from a terminal to the opposite side of the channel. Also let the *flow* across a cut χ be the number of nets that must cross χ, namely those nets having terminals on both sides of χ and those having an endpoint of χ as a terminal. The *capacity* of χ is one greater than the maximum of the horizontal and vertical separations of its endpoints; if χ is the line segment from (x_1, y_1) to (x_2, y_2), then

$$capacity(\chi) = max\{|x_1 - x_2|, |y_1 - y_2|\} + 1 \ .$$

The cut χ is *safe* if $flow(\chi) \leq capacity(\chi)$, which means that there is enough space along χ for the wires to get through. Since we are restricting attention to channels verified to be routable in one layer, as mentioned in Section 1, the following lemma holds.

Lemma 2.1 *The following three statements are equivalent:*
1. The channel is routable.
2. Every critical cut is safe.
3. Every pivotal cut is safe.

Proof. This lemma follows from the corresponding results in [16, §2.1,2.3,2.6.5]. Our slightly different definitions of flow and capacity allow Maley's formulation in terms of cuts emanating from "feature" endpoints to correspond to cuts emanating from terminals. Since we allow routing on the channel boundaries, the only "features" are terminals and two routing obstacles (horizontal lines) located one unit outside of what we have been referring to as the channel boundaries. □

We can further strengthen the result for critical cuts as follows. Define the *span* of a cut χ to be the horizontal distance between its endpoints. Call χ *sparse* if χ is not vertical and $flow(\chi) \leq span(\chi) + 1$, and *dense* otherwise. A sparse cut is safe regardless of the separation, but a dense cut χ is safe if and only if the separation is at least $flow(\chi) - 1$.

Lemma 2.2 *The minimum channel separation is the maximum of* $flow(\chi) - 1$ *over dense critical cuts* χ. □

When all single-sided nets are on the bottom, we can strengthen the result for pivotal cuts, but first we must review results regarding *contours* of single-sided nets.

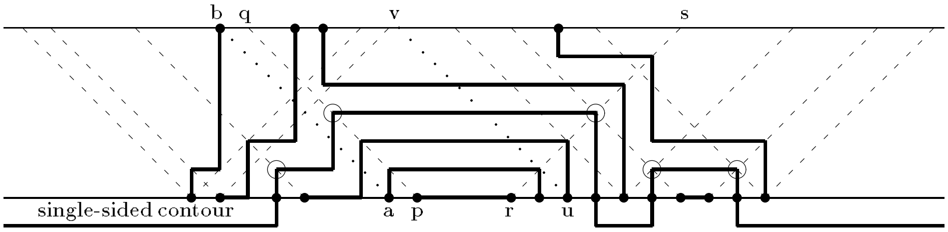

Fig. 1: With all single-sided nets on the bottom, only those 45° cuts from bottom terminals shown with dashed lines need to be checked (Lemma 2.4). The hollow circles mark the convex corners of the contour of single-sided nets, which runs unit distance below the channel where no single-sided nets are present and otherwise follows the single-sided net protruding farthest into the channel.

The *contour* of the single-sided nets is the routing boundary that the two-sided nets must stay unit distance away from when the single-sided nets are routed as tightly as possible against the bottom of the channel. (See Figure 1.)

The following Lemma from [7] shows that a contour of single-sided nets can be found efficiently.

Lemma 2.3 *The bendpoints in the contour of a set of N single-sided nets can be found in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg\lg N})$) time using $O(\frac{N}{\lg N})$ (resp. $O(\frac{N\lg\lg N}{\lg N})$) processors on a CREW (CRCW) PRAM.* $\Box$

Now, we are ready to state a result of [12] relating to pivotal cuts:

Lemma 2.4 *A channel with all single-sided nets on the bottom is routable if and only if all 45° cuts from bottom terminals of two-sided nets and all 45° cuts crossing the contour of single-sided nets at a convex corner are safe.* $\Box$

3 The Algorithms

3.1 The Minimum Separation Problem

Our algorithm for this problem is based on Lemma 2.2. To ensure that vertical cuts are captured, first add a dummy terminal across from each real terminal. Then we find the minimum separation that makes all dense (critical) cuts emanating from bottom terminals safe. To find all dense cuts emanating from b_j, we search for the two farthest dense cuts, one going to the right and one going to the left from b_j; these two cuts form a "cone" such that cuts emanating from b_j are dense if and only if they lie inside the cone.

We now provide some further definitions and notations used in this subsection. First, we say that a terminal is covered by a single-sided net on its side of the channel if it lies in the closed interval defined by the endpoints of the net; define $S(\tau)$ to be the number of single-sided nets covering τ. Now, two-sided nets are said to lie to the left or right of a terminal on the top (or bottom) according to the location of the net's top (respectively bottom) terminal. Also, a two-sided net is a right net if its top terminal is to the right of its bottom terminal; it is a left net if its top terminal is to the left of its bottom terminal. Define $R(\tau)$ to be the number of right nets to the left of terminal τ, and $L(\tau)$ to be the number of left nets to the left of τ. Also define $\mathrm{IL}(\tau)$ (and $\mathrm{IR}(\tau)$) to be 1 if τ is a terminal of a left (respectively right) net, and zero otherwise.

The heart of this algorithm is to form the cone for each bottom terminal b_j. According to the definition, a nonvertical cut $\overline{t_i b_j}$ is dense if $|t_i - b_j| + 1 < flow(\overline{t_i b_j})$. We now show how to find the farthest cuts emanating from each terminal b_j on the bottom that are dense. Note that for any dense cut $\overline{t_i b_j}$, $R(t_i) \leq R(b_j)$, and $L(t_i) \geq L(b_j)$. Thus, for dense cuts, $flow(\overline{t_i b_j}) = L(t_i) - L(b_j) + \mathrm{IL}(t_i) + R(b_j) - R(t_i) + \mathrm{IR}(b_j) + S(t_i) + S(b_j)$. Defining $\mathrm{bl}(j)$ to be $b_j + L(b_j) - R(b_j) - S(b_j) - \mathrm{IR}(b_j) + 1$ and $\mathrm{tl}(i)$ to be $t_i + L(t_i) - R(t_i) + S(t_i) + \mathrm{IL}(t_i)$, we need to find the smallest t_i such that $\mathrm{bl}(j) < \mathrm{tl}(i)$; for similar definitions of $\mathrm{tr}(i)$ and $\mathrm{br}(j)$, we also find the largest t_i such that $\mathrm{tr}(i) < \mathrm{br}(j)$. It can be shown that the four functions bl, tl, br, and tr are non-decreasing. Now, we can give an algorithm for forming the cone for each bottom terminal.

procedure FIND-CONES

1. Compute $\mathrm{tl}(i), \mathrm{bl}(j), \mathrm{tr}(i)$ and $\mathrm{br}(j)$ for $1 \leq i \leq m$ and $1 \leq j \leq n$.
2. Merge $\mathrm{tl}(i)$ with $\mathrm{bl}(j)$ and $\mathrm{tr}(i)$ with $\mathrm{br}(j)$ in order of nondecreasing values. If a tie occurs, put $\mathrm{br}(j)$ before $\mathrm{tr}(i)$ and put $\mathrm{bl}(j)$ after $\mathrm{tl}(i)$.
3. For each $\mathrm{bl}(j)$, find the nearest $\mathrm{tl}(i)$ to the right in the merged sequence. If we do not find such a $\mathrm{tl}(i)$ corresponding to a t_i with lesser x-coordinate than b_j, then the farthest dense cut to the left from b_j is vertical. Similarly, for each $\mathrm{br}(j)$, find the nearest $\mathrm{tr}(i)$ to the left in the merged sequence, and select a vertical cut if necessary.

The merging can be done using the approach of Kruskal [14] in $O(\lg\lg N)$ time with $O(N)$ work on a CREW PRAM. Also steps 1 and 3 can be implemented by using prefix-sums. Therefore, algorithm FIND-CONES can be implemented with optimal work in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg\lg N})$) time on a CREW (CRCW) PRAM.

Once we find the cone for each bottom terminal, we can use the information to find the minimum separation for the single-layer channel routing problem:

procedure MINIMUM-SEPARATION

1. Apply algorithm FIND-CONES to find the farthest dense cuts to form a cone for every terminal on the bottom.
2. Find the maximum flow $F(b_j)$ among the cuts inside the cone for every terminal b_j.
3. The minimum separation is $-1 + \max\{F(b_1), F(b_2), \ldots, F(b_n)\}$.

Theorem 3.1 *Algorithm* MINIMUM-SEPARATION *finds the minimum separation for single-layer channel routing with $O(N)$ work in time $O(\lg N)$ on a CREW PRAM and in time $O(\frac{\lg N}{\lg\lg N})$ on a CRCW PRAM.*

Proof. We have already explained how step 1 can be performed within the specified time and processor bounds, and step 3 simply involves a minimum that can be computed in the same bounds. To find the maximum flow in each cone in step 2, we use the range maxima technique. Since the flow for a dense cut $\overline{t_i b_j}$ is $L(t_i) - L(b_j) + \mathrm{IL}(t_i) + R(b_j) - R(t_i) + \mathrm{IR}(b_j) + S(t_i) + S(b_j)$, and the terms dependent on j are fixed for any given cone, the task is to find the maximum of $\mathrm{tl}(i) - t_i$ over each cone. The preprocessing for range maxima and the single query per b_j can also be implemented within the stated bounds. □

3.2 The Offset Range Problem

In this subsection, we consider the offset range problem for single-layer channel routing with single-sided nets on one side. (We restrict attention here to the "intermediate" rather than the "general" model defined in Section 1, because there could be more than linearly many feasible offsets when there are single-sided nets on both sides of the channel [11].) Without loss of generality, assume that all single-sided nets are on the bottom. Henceforth, we use s to represent the separation and d for the offset (the positive or negative distance by which the upper block of terminals is moved right from its original position).

According to Lemma 2.4, we only need to ensure that all 45° cuts from bottom terminals of two-sided nets and all 45° cuts crossing the contour of single-sided nets at a convex corner are safe. The most direct approach to obtain the desired result for offset range is to simply check *all* 45° cuts from bottom terminals, but we use a slightly less direct approach that will help us to solve the optimal offset problem as well. For this approach, we need to separate the bottom terminals of two-sided nets from those of single-sided nets. Let $a_1, a_2, \ldots, a_m$ be the x-coordinates of the bottom terminals of two-sided nets and $c_1, c_2, \ldots, c_h$ the x-coordinates of the bendpoints of the contour of single-sided nets.

Using Lemma 2.4, it is shown in [12] that we need only check $4m$ cuts. To obtain this formulation, we define T_i to be the number of two-sided nets whose bottom terminals are to the left of c_i. Also, let E_i be the extension of the single-sided contour at c_i, the nonnegative distance that the contour rises above its baseline at that column. Finally, define c_{r_i} to be the nearest bendpoint of the single-sided contour to the right of a_i such that $T_{r_i} - i + E_{r_i} > s$ and c_{l_i} to be the nearest bendpoint to the left of a_i such that $i - T_{l_i} + E_{l_i} > s + 1$.

Adapting the result of [12] to the notation in this paper, we have that the pair (s, d) is feasible if and only if $l(s) < d < u(s)$, where

$$l(s) = \max_{1 \le i \le m} \{a_{i-s-1}, c_{l_i} - E_{l_i}\} + s - t_i \tag{1}$$

and

$$u(s) = \min_{1 \le i \le m} \{a_{i+s+1}, c_{r_i} + E_{r_i}\} - s - t_i\ . \tag{2}$$

Here we define $a_j = -\infty$ if $j \le 0$ and $a_j = \infty$ if $j > m$; also c_{l_i} (c_{r_i}) is defined to be $-\infty$ (∞) if there is no bendpoint satisfying the necessary conditions.

We can now prove the following theorem:

Theorem 3.2 *The offset range for single-layer channel routing with single-sided nets on one side can be found with $O(N)$ work in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg\lg N})$) time on a CREW (CRCW) PRAM.*

Proof. The contour of single-sided nets can be found within the requisite resource bounds by Lemma 2.3. The l_i and r_i values can be found by solving an ANSV problem, and then we need only compute a maximum and minimum of $m \leq N$ values; these operations can be performed with $O(m)$ work in $O(\lg m)$ time on the CREW or $O(\lg\lg m)$ time on the CRCW as indicated in Section 2. □

3.3 The Optimal Offset Problem

This subsection considers the optimal offset problem for river routing and channels with single-sided nets on one side (the bottom). In both cases, the optimal offset problem is solved by using an algorithm for offset range as a subroutine. For channels with single-sided nets on one side, we use the results of Section 3.2 to obtain optimal offset results better than those of Aggarwal and Park, even though their results apply only to river routing. For river routing on the CRCW, we improve their optimal offset bounds even further.

Our algorithm adapts Mirzaian's halving technique [17] for relating optimal offset to offset range. 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 offset range 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 contour of single-sided nets, remove every other two sided net, and compact the channel horizontally to eliminate the freed space. More precisely, we perform the transformation specified as follows:

$$t_i^e = t_{2i} - i\ , \quad a_i^e = a_{2i} - i\ , \quad r_i^e = r_{2i}\ , \quad \text{and} \quad l_i^e = l_{2i} \qquad 1 \leq i \leq \lfloor m/2 \rfloor$$

and

$$c_j^e = c_j - \lceil T_j/2 \rceil\ , \quad \text{and} \quad E_j^e = \lfloor E_j/2 \rfloor \qquad j \in \{\, r_i^e, l_i^e \mid 1 \leq i \leq \lfloor m/2 \rfloor \,\}\ .$$

The following lemma, from [12], relates optsep(P) to optsep(P^e):

Lemma 3.1 *Let* $s = \text{optsep}(P)$ *and* $s^e = \text{optsep}(P^e)$*. Then* $2s^e \leq s \leq 2s^e + 3$*.* □

Let P_0 be the original problem, and define P_k to be P_{k-1}^e, for $1 \leq k \leq p = \lg m$. (For ease of presentation, we assume the number of two-sided nets m is 2^p for some integer p.) Also, let $s_k = \text{optsep}(P_k)$. From the above lemma, once we know s_{k+1}, then s_k can be solved by checking only 4 possible separations to achieve the optimal offset. Now, s_{k-1} can also be solved by checking all the possible separations derived from the 4 separations. (Again each possible separation of s_k induces 4 possible separations for s_{k-1}.) By considering the union of these separations, we only have to check 10 possible separations to solve s_{k-1}. Continuing this line of reasoning, we have the following corollary:

Corollary 1 *If* s_k *is known, then* s_{k-l} *can be solved by checking only* $3 \cdot 2^l - 2$ *possible separations.* □

Figure 2 shows our algorithm to solve the optimal offset problem. The algorithm finds the optimal separations $s_{p/2^i}$, for $1 \leq i \leq \lg p$. Each separation is determined from previously computed separations by using Corollary 1.

We first derive the resource bounds of this algorithm when applied to river routing, and then we extend the analysis to channels with single-sided nets on one side.

```
  procedure OPTIMAL-SEPARATION
1 for i ← lg p to 1 do
2     find P_{p/2^i}
3 endfor
4 for i ← 1 to lg p do
5     find s_{p/2^i}
6 endfor
7 find s_0
```

Fig. 2: This algorithm finds the minimum separation for the optimal offset problem.

Theorem 3.3 *The optimal offset for river routing can be found with $O(N \lg\lg N)$ work in $O(\lg N \lg\lg N)$ (respectively $O((\lg\lg N)^2)$) time on a CREW (CRCW) PRAM.*

Proof. First we explain the CREW result. Lines 1–3 can be executed in time $O(\lg m)$ time with $O(m)$ work. (To see this, it suffices to consider repeated halving of the number of two-sided nets in constant time, with the work at each stage decreasing geometrically to a constant as the number of nets decreases.) In lines 4–6, each $s_{p/2^i}$ can be found in $O(p) = O(\lg m)$ time as follows. First, $s_{p/2}$ has at most $2^{p/2}$ possible values because there are only $2^{p/2}$ nets in $P_{p/2}$. The feasibility of each separation can be checked in $O(p)$ time with $O(2^{p/2})$ work by the proof of Theorem 3.2 or [1]. So all the possible separations can be checked simultaneously with $O(2^p)$ work. The minimum separation among the feasible separations is the optimal separation. Now, suppose $s_{p/2^i}$ is known; then at most $O(2^{p/2^{i+1}})$ possible values need to be checked for $s_{p/2^{i+1}}$ by Corollary 1. Using the same reasoning as above, each possible separation can be checked in $O(p)$ time with $O(2^p/2^{p/2^{i+1}})$ work, and $s_{p/2^{i+1}}$ can be found in $O(p)$ time with $O(2^p)$ work. Finally, Line 7 is again $O(p)$ time and $O(2^p)$ work. The total time and work, including all the passes through the loop in lines 4–6, are $O(p \lg p)$ and $O(2^p \lg p)$, respectively.

On the CRCW, we can do each pass through the loop in lines 1–3 in $O(\lg\lg m)$ time with $O(m)$ work, and there are $O(\lg\lg m)$ passes. The remaining analysis is the same as before except with the $O(\lg\lg m)$ time offset range result for river routing on the CRCW that comes from Aggarwal and Park [1] or the proof of Theorem 3.2. This yields $O(\lg p)$ time and $O(2^p)$ work to compute each $s_{p/2^i}$ (and s_0). $\square$

When there are single-sided nets on one side of the channel, we need modify our argument only slightly. The analysis above remains valid, except that we must do one computation up front of the contours and the l_i and r_i values. Adding in the extra resources required, we obtain the following:

Corollary 2 *The optimal offset problem for channels with single-sided nets on one side can be solved with $O(N \lg\lg N)$ work in $O(\lg N \lg\lg N)$ (respectively $O(\frac{\lg N}{\lg\lg N})$) time on a CREW (CRCW) PRAM.* $\square$

4 Conclusion

This paper has provided efficient parallel algorithms for (1) the minimum separation problem for general single-layer channels and (2) offset problems for single-layer channels in which only one side of the channel has multiple connections to a single net. In addition, we have improved previous results for optimal offset in river routing problems, where each net has exactly one terminal on each side of the channel.

The above results also apply whenever a channel boundary that is allowed to have multiple connections to a single net is also allowed to be ragged. (We must first add dummy terminals at all the bendpoints of the ragged boundary, increasing N accordingly.) For the minimum separation problem, we need only incorporate the extension of the boundary at t_i into the range maxima computation in the proof of Theorem 3.1 and then the extension at b_j into the maximization over the dense cuts emanating from b_j. For the offset problems, we simply incorporate boundary extension terms into each of the a and c terms in Equations (1) and (2).

An obvious open question is whether any of the bounds on time or work can be improved. In particular, the algorithms for optimal offset use $N \lg\lg N$ work rather than the $O(N)$ work that can be achieved sequentially. An additional open question is whether offset problems can be efficiently solved in parallel when both sides of the channel contain single-sided nets; for sequential computation, this problem is considered in [11].

Acknowledgements

This work was supported in part by NSF grants CCR-9109550 and CCR-9321388. Thanks also to Uzi Vishkin and Joseph JáJá of the University of Maryland, Omer Berkman of King's College, and Yossi Matias of AT&T Bell Labs for helpful discussions.

References

1. A. Aggarwal and J. K. Park. Parallel searching in multidimensional monotone arrays. Research Report RC 14826, IBM T. J. Watson Research Center, Yorktown Heights, NY, Aug. 1989. Earlier version in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*.
2. S. Albers and T. Hagerup. Improved parallel integer sorting without concurrent writing. *Information and Computation*, 1997. To appear. Earlier version in Proceedings of the 3rd Annual SIAM Symposium on Discrete Algorithms.
3. H. Bast and T. Hagerup. Fast parallel space allocation, estimation and integer sorting. *Information and Computation*, 123(1):72–110, 1995.
4. O. Berkman, P. B. Gibbons, and Y. Matias. On the power of randomization for the common PRAM. In *Proceedings of 3rd Annual Israel Symposium on Theory of Computing and Systems*, pages 229–240, 1995.
5. O. Berkman, B. Schieber, and U. Vishkin. Optimal doubly logarithmic parallel algorithms based on finding all nearest smaller values. *Journal of Algorithms*, 14(3):344–370, May 1993.
6. O. Berkman and U. Vishkin. Recursive star-tree parallel data-structure. *SIAM Journal on Computing*, 22(2):221–242, Apr. 1993.
7. S.-C. Chang, J. JáJá, and K. W. Ryu. Optimal parallel algorithms for one-layer routing. Technical Report UMIACS-TR-89-46, University of Maryland Institute for

Advanced Computer Studies, Apr. 1989.

8. R. Cole and U. Vishkin. Faster optimal prefix sums and list ranking. *Information and Control*, 81:334–352, 1989.
9. D. Dolev, K. Karplus, A. Siegel, A. Strong, and J. D. Ullman. Optimal algorithms for structural assembly. *VLSI Design*, pages 38–43, 1982. Earlier version in *Proceedings of the 13th ACM Symposium on Theory of Computing.*
10. R. I. Greenberg and F. M. Maley. Minimum separation for single-layer channel routing. *Information Processing Letters*, 43(4):201–205, Sept. 1992.
11. R. I. Greenberg and J.-D. Shih. Feasible offset and optimal offset for general single-layer channel routing. *SIAM Journal on Discrete Mathematics*, 8(4):543–554, Nov. 1995.
12. R. I. Greenberg and J.-D. Shih. Single-layer channel routing and placement with single-sided nets. *Computers & Mathematics With Applications*, 32(4):1–7, 1996.
13. J. JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.
14. C. P. Kruskal. Searching, merging and sorting in parallel computation. *IEEE Trans. Computers*, C-32(10):942–946, Oct. 1983.
15. R. E. Ladner and M. J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, Oct. 1980.
16. F. M. Maley. *Single-Layer Wire Routing and Compaction.* MIT Press, 1990.
17. A. Mirzaian. River routing in VLSI. *Journal of Computer and System Sciences*, 34:43–54, 1987.
18. S. Rajasekaran and S. Sen. On parallel integer sorting. *Acta Informatica*, 29(1):1–15, 1992.
19. S. Saxena. Parallel integer sorting and simulation amongst CRCW models. *Acta Informatica*, 33(7):607–619, 1996.