A New Staircase Separator Theorem*

Viet Hai Nguyen

Database Laboratory, Department of Computer Science,
Swiss Federal Institute of Technology (EPF) Lausanne,
IN - Ecublens, 1015 Lausanne, Switzerland
VietHai.Nguyen@di.epfl.ch

Abstract. The notion of staircase separator, introduced in [2], greatly facilitates the design of divide-and-conquer algorithms for problems on rectangles. We generalize the concept of staircase separator to *k-perfect staircase separator*, namely a set of staircase separators which partitions a set $\mathcal{S}$ of n axis-parallel, disjoint rectangles into k subsets of (almost) equal size. We derive an optimal $O(\log n)$ time parallel algorithm for computing a k-perfect staircase separator, using $O(n)$ processors on the CREW PRAM model of computation. For a special case, where $k = 2$, this result provides a new bound of $\lceil \frac{n}{2} \rceil$, in compared to $\lceil \frac{7n}{8} \rceil$ in [2], on the quality of staircase separators for sets of rectangles.

1 Introduction

In this paper we consider the problem of computing a staircase separator for a set $\mathcal{S}$ of n axis-parallel, disjoint rectangles in the plane. The concept of staircase separator has been shown to be very helpful in solving exactly or approximately the rectilinear shortest path problem (for examples, see [2, 3, 5, 8]). The first staircase separator theorem was proved by Atallah and Chen. In [2], the authors proved that there is a staircase separator which partitions $\mathcal{S}$ into two subsets such that each subset contains at most $\lceil \frac{7n}{8} \rceil$ rectangles and this separator can be constructed in $O(\log n)$ time, using $O(n)$ processors on the CREW PRAM model of computation. The authors proposed a divide and conquer approach for solving the rectilinear shortest path problem by using staircase separators to recursively partition the set of rectangles. Other divide and conquer algorithms using the result on staircase separators can be found in, for example, [5, 8].

We generalize the concept of staircase separator to *k-perfect staircase separator* (a precise definition is given in the next section) and present an efficient parallel algorithm for computing these staircase separators, using an approach that is quite different from that of [2]. More precisely,

* work done while at Departement Informatik, ETH Zürich, Switzerland.

we present an optimal $O(\log n)$ time parallel algorithm for computing a k-perfect staircase separator, using $O(n)$ processors on the CREW PRAM model of computation. Letting $k = 2$, this result implies an optimal bound of $\lceil \frac{n}{2} \rceil$ on the quality of staircase separators for sets of rectangles, thus considerably improving the previous bound of $\lceil \frac{7n}{8} \rceil$ derived by Atallah and Chen in [2].

Our results make use of simple geometric observations and classical data structuring techniques. The visibility graph structure for a set of rectangles in the plane introduced in this paper is used as a central tool for computing perfect staircase separators. We hope that the results and techniques presented here may also be useful for solving other geometric problems on sets of rectangles in the plane.

Due to space limitation, some proofs are omitted. More details and further results on staircase separators can be found in [9].

2 Preliminaries

Recall that, a *rectilinear path* π_{st} from s to t is a sequence of points $(p_0, \ldots, p_m)$, where $p_0 = s$, $p_m = t$, and p_i and p_{i+1} differ in exactly one coordinate. π_{st} is called *x-monotone* (*y-monotone*) if the x-coordinates (y-coordinates) of points along the directed path from s to t are monotone, either non-decreasing or non-increasing. A path π is called a *staircase* if it is xy-monotone. A staircase is *unbounded* if it starts and ends with a semi-infinite segment, i.e. a segment that extends to infinity on one side. (By convention, a staircase $\pi = (p_0, \ldots, p_m)$ starts on its left and ends on its right, i.e. p_0 and p_m are the leftmost and rightmost points of π, respectively.) Staircases can be *increasing* or *decreasing*, depending on whether they are y-monotone increasing or decreasing. Let $\mathcal{S}$ be a set of n (axis-parallel) non-overlapping rectangles in the plane. For the sake of simplicity, we assume that the rectangles of $\mathcal{S}$ are in general position. A staircase which does not intersect the interior of any rectangle of $\mathcal{S}$ is called a *staircase separator*.

A point p in a set $\mathcal{V}$ of points in the plane is called a *NW-maximum* if there exists no q in $\mathcal{V}$ that lies northwest of p. Formally, a point $p = (x_1, y_1)$ in $\mathcal{V}$ is called a *NW-maximum* if there exists no $q = (x_2, y_2)$ in $\mathcal{V}$ such that $q \neq p$ and $x_2 \leq x_1$ and $y_2 \geq y_1$. *XY-maximum* can similarly be defined, for any $X \in \{N, S\}$ and $Y \in \{E, W\}$.

A point p is *above* (resp., to the *left* of) an unbounded staircase $\mathcal{C}$ iff (i) the vertical (resp., *horizontal*) line $l(p)$ incident with p intersects $\mathcal{C}$ and (ii) for any point $p' \in l(p) \bigcap \mathcal{C}$, p is above (resp., to the *left* of) p'.

A rectangle R is *above* (resp., to the *left* of) an unbounded staircase $\mathcal{C}$ iff any point of R is above (resp., to the left of) $\mathcal{C}$; we can equivalently say that $\mathcal{C}$ is *below* (resp., to the *right* of) R. A rectangle R_1 is *above* (resp., to the *left* of) a rectangle R_2 iff for any two points $p_1 = (x_1, y_1)$ of R_1 and $p_2 = (x_2, y_2)$ of R_2, $y_1 > y_2$ (resp., $x_1 < x_2$); we can equivalently say that R_2 is *below* (resp., to the *right* of) R_1.

Let $\mathcal{S}$ be a set of n rectangular obstacles in the plane. $\mathcal{V}_{\mathcal{S}}$ denotes the set of obstacle vertices. Let R be a rectangle in $\mathcal{S}$. Denote R's corners by R_{XY} (or R_{YX}), for any $X \in \{N, S\}$ and $Y \in \{E, W\}$ (e.g. the northwest corner of R is denoted by R_{NW} or R_{WN}). Let $NW(R)$ denote the first rectangle of $\mathcal{S}$ that is hit by a vertical ray going upward from the northwest corner R_{NW} of R. If the vertical ray goes to infinity without hitting a rectangle of $\mathcal{S}$, then $NW(R)$ is defined to be a rectangle ∞_N at infinity. Similarly, we define $XY(R)$ ($YX(R)$, respectively) for any $X \in \{N, S\}$ and $Y \in \{E, W\}$ and call these rectangles *neighbours* of R. R is *vertically (horizontally) visible* from a staircase π iff (i) there is a vertical (horizontal) line segment s connecting two interior points $p_1 \in R$ and $p_2 \in \pi$; (ii) for any such line segment s, s does not intersect the interior of any other obstacle. Similarly, we define a visibility relationship between the rectangles of $\mathcal{S}$.

Definition 1. Two rectangles R_1 and R_2 are *vertically (horizontally) visible* iff (i) there is a vertical (horizontal) line segment s connecting two (interior) points $p_1 \in R_1$ and $p_2 \in R_2$; (ii) for any such line segment s, s does not intersect the interior of any other obstacle.

The following definition describes important properties of a k-perfect staircase separator:

Definition 2. For an integer k, $2 \leq k \leq n$, a set of (unbounded and bounded) staircases $\mathcal{C}$ is called a *k-perfect staircase separator* of a set $\mathcal{S}$ of n disjoint rectangles if it satisfies the following properties:

1) For all $C \in \mathcal{C}$, C does not intersect the interior of any rectangle in $\mathcal{S}$.
2) $\mathcal{C}$ partitions $\mathcal{S}$ into k subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that each of $\mathcal{S}_i$, $1 \leq i \leq k$ contains no more than $\lceil \frac{n}{k} \rceil$ rectangles in $\mathcal{S}$.
3) $\mathcal{C}$ consists of at most n segments.

We shall use the plane-sweep tree technique [1, 4] (see also [6]). The following known result will allow us to avoid describing in detail the plane-sweep tree data structure, and thus to concentrate on the presentation of our algorithm for constructing the graph structure in Section 3.

Lemma 3. *([4]) Given a set of n nonintersecting horizontal segments in the plane, the problem of determining, for each endpoint p, the closest segment directly below p can be solved in $O(\log n)$ time, using a total of $O(n \log n)$ operations.*

The following definition is borrowed from [2]. For a subset $\mathcal{S}'$ of $\mathcal{S}$, consider the set of all increasing unbounded staircases that are above all rectangles in $\mathcal{S}'$. From the set of all such staircases choose the *lowest-rightmost* one; that is, if $\mathcal{C}$ is the chosen one, then there is no unbounded staircase $\mathcal{C}'$ above $\mathcal{S}'$ with a point of $\mathcal{C}'$ strictly below or to the right of a point of $\mathcal{C}$. Denote such a $\mathcal{C}$ by $Max_{NW}(\mathcal{S}')$. We can similarly define $Max_{XY}(\mathcal{S}')$ for any $X \in \{N, S\}$ and $Y \in \{E, W\}$.

3 Computing Perfect Staircase Separators in Parallel

In this section we present an optimal parallel algorithm for computing perfect staircase separators for a set $\mathcal{S}$ of n rectangular obstacles in the plane. We begin with an algorithm for computing a 2-perfect staircase separator (which we simply call a *perfect staircase separator*). Let $\mathcal{C}$ be an unbounded increasing staircase, which begins with a horizontal semi-infinite segment and ends with a vertical semi-infinite segment. Given a rectangle $R \in \mathcal{S}$ such that R is above $\mathcal{C}$, we draw two horizontal line segments h_1, h_2 and one vertical line segment v from R_{NE} and R_{SE}, the northeast and southeast corners of R, to $\mathcal{C}$ (see Fig. 1). Let $E_{\mathcal{C}}(R)$ ($SE_{\mathcal{C}}(R)$, resp.) denote the *open* rectilinear region bounded by the east edge of R, h_1, h_2 and $\mathcal{C}$ (h_2, v and $\mathcal{C}$, resp.). We have the following lemma.

Lemma 4. *Let L be the set of all rectangles in $\mathcal{S}$ which are vertically visible from a staircase separator $\mathcal{C}$, R be the rightmost [1] rectangle in L. The following assertions hold*

(1) No rectangle in $\mathcal{S}$ intersects $SE_{\mathcal{C}}(R)$;
(2) No rectangle in $\mathcal{S}$ intersects $E_{\mathcal{C}}(R)$, i.e. R is horizontally visible from $\mathcal{C}$.

Now, let $\mathcal{C}_0$ be an arbitrary unbounded increasing staircase separator, which begins with a horizontal semi-infinite segment and ends with a vertical semi-infinite segment. $\mathcal{C}_0$ partitions $\mathcal{S}$ into $\mathcal{S}_l$ and $\mathcal{S}_r$, where $\mathcal{S}_l$ is on the left and $\mathcal{S}_r$ is on the other side of $\mathcal{C}$. Assuming $\mathcal{S}_l$ is not empty, we define a *(left) bending* operation as follows. Among all rectangles of $\mathcal{S}_l$

[1] Recall that R' is to the right of R iff for any two points $p' \in R'$ and $p \in R$, $p'.x \geq p.x$.

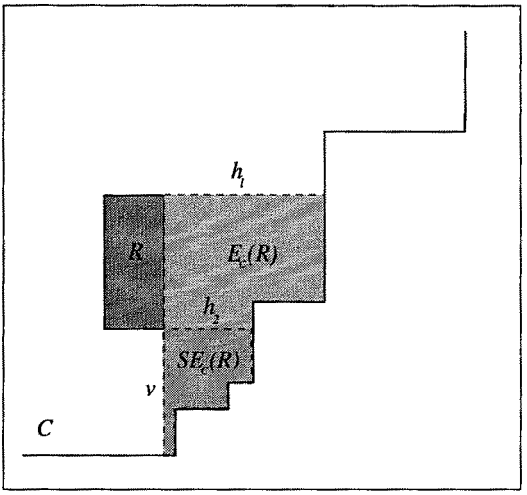

Fig. 1. Illustration for Lemma 4

which are vertically visible from $\mathcal{C}_0$, select the *rightmost* rectangle R. R is horizontally visible from $\mathcal{C}_0$, by Lemma 4. Let p_1 and p_2 be the vertical and horizontal projection points of $p = R_{NW}$ onto $\mathcal{C}_0$. Now we replace the rectilinear subpath of $\mathcal{C}_0$, which starts at p_1 and ends at p_2, by a new rectilinear subpath $\overline{p_1 p} \bigcup \overline{p p_2}$. In other words, we "bend" the edges of $\mathcal{C}_0$ around R to obtain a new staircase separator $\mathcal{C}_1$. Obviously, $\mathcal{C}_1$ partitions $\mathcal{S}$ into $\mathcal{S}'_l$ and $\mathcal{S}'_r$, where $|\mathcal{S}'_l| = |\mathcal{S}_l| - 1$, $|\mathcal{S}'_r| = |\mathcal{S}_r| + 1$. Starting with an arbitrary unbounded staircase separator with $|\mathcal{S}_r| < \lceil \frac{n}{2} \rceil$, by applying successively the above described bending operation at most $\lfloor \frac{n}{2} \rfloor$ times, we can obtain a perfect staircase separator $\mathcal{C}$.

The idea of our parallel algorithm for constructing a perfect staircase separator $\mathcal{C}$, which partitions $\mathcal{S}$ into two subsets $\mathcal{S}_l$ and $\mathcal{S}_r$, is to compute the subset $\mathcal{S}_r$ and then construct its northwest boundary $Max_{NW}(\mathcal{S}_r)$. Recall that a point p in $\mathcal{V}$ is called a *NW-maximum* if there exists no q in $\mathcal{V}$ that lies northwest of p. $Max_{NW}(\mathcal{S})$ can be constructed in constant time, using $O(n)$ operations from a sorted set of all NW-maxima of $\mathcal{V}_{\mathcal{S}}$. The latter set can be computed in $O(\log n)$ time, using $O(n \log n)$ operations by applying the cascading divide and conquer approach for solving the dominance counting problem [4] (see also [6]). Now we show how to solve the other task by first computing a graph structure $\mathcal{G}_{\mathcal{R}}(\mathcal{S}) = < \mathcal{S}, \mathcal{R} >$, where $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ and $(R_1, R_2) \in \mathcal{R}$ iff R_1 and R_2 are vertically visible.

$\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ can be implemented by a data structure $\mathcal{D}(\mathcal{G}_{\mathcal{R}}(\mathcal{S}))$ as follows. The edges of $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ are simply implemented by using two-way pointers.

Let $Nneighbour_{\mathcal{R}}(R)$ denote the set of rectangles vertically visible from the north edge of R, sorted from left to right. $Zneighbour_{\mathcal{R}}(R)$ can similarly be defined, for any $Z \in \sigma$. For the current goal, we need to store at each node R in the graph $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ a sorted set $Nneighbour_{\mathcal{R}}(R)$ ($Sneighbour_{\mathcal{R}}(R)$, resp.), which allows us to list all north (south, resp.) neighbours of R, from left to right. It is not difficult to see that the resulting data structure $\mathcal{D}(\mathcal{G}_{\mathcal{R}}(\mathcal{S}))$ has size $O(n)$.

Now, let $Nneighbour_{\mathcal{V}}(R)$ denote the set of obstacle vertices vertically visible from the north edge of R, sorted from left to right. Similarly, let $Zneighbour_{\mathcal{V}}(R)$ denote the sorted set of obstacle vertices visible from the Z-edge of R, for any $Z \in \sigma$. The following simple geometric observations are essential for the construction of the graph $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$.

Lemma 5. *Let R and R' be two nonintersecting rectangles, where R is above R'. R and R' are vertically visible iff one of the following conditions holds*

(i) $Nneighbour_{\mathcal{V}}(R') = \emptyset$, *and* R'_{NW} *and* R'_{NE} *are in* $Sneighbour_{\mathcal{V}}(R)$,
(ii) $Sneighbour_{\mathcal{V}}(R) = \emptyset$, *and* R_{SW} *and* R'_{SE} *are in* $Nneighbour_{\mathcal{V}}(R')$,
(iii) R'_{NW} *is the last element in* $Sneighbour_{\mathcal{V}}(R)$ *and* R_{SE} *is the first element in* $Nneighbour_{\mathcal{V}}(R')$,
(iv) R'_{NE} *is the first element in* $Sneighbour_{\mathcal{V}}(R)$ *and* R_{SW} *is the last element in* $Nneighbour_{\mathcal{V}}(R')$.

Now, using the results of Lemma 3 and Lemma 5 we can prove the following theorem.

Theorem 6. *Given a set $\mathcal{S}$ of n rectangular obstacles in the plane, the graph $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ can be constructed in $O(\log n)$ time, using a total of $O(n \log n)$ operations.*

Having preprocessed $\mathcal{S}$ into $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$, we now turn back to the computation of $\mathcal{S}_r$. We shall employ the Euler-Tour technique for tree computation [10]. However, the (undirected) graph $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ may not be a tree: from a node R to a node R', $R' \neq R$, there may be more than one path, i.e. there may be a "cycle" from R to R'. Therefore, in order to apply successfully the Euler-Tour technique, we need to transform the graph $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ into a tree structure $\mathcal{T}_{\mathcal{R}}(\mathcal{S}) = < \mathcal{S}, \mathcal{R}^* >$, where $\mathcal{R}^* \subset \mathcal{R}$. $\mathcal{T}_{\mathcal{R}}(\mathcal{S})$ is copied from $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ by a cycle-elimination procedure which costs $O(1)$ time with $O(n)$ operations as follows. For each node R of $\mathcal{G}_{\mathcal{R}}(\mathcal{S})$ with $|Nneighbour_{\mathcal{R}}(R)| > 1$, only the edge $(R', R) \in \mathcal{R}$ connecting R to the rightmost rectangle R' above R, is retained in $\mathcal{R}^*$. Moreover, the node

set of $\mathcal{T}_{\mathcal{R}}(\mathcal{S})$ can easily be organized into *adjacency lists* with additional pointers [6] in $O(\log n)$ time, using $O(n)$ operations.

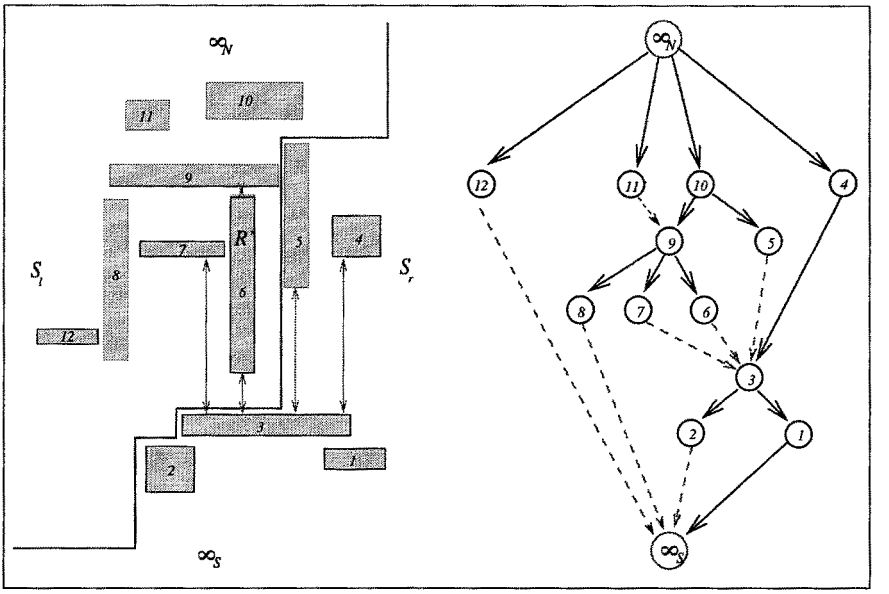

Fig. 2. $\mathcal{T}_{\mathcal{R}}(\mathcal{S})$

Observe that the order in which the rightmost vertically (from a staircase separator $\mathcal{C}$) visible rectangle R is included into $\mathcal{S}_r$ is exactly the reverse of the preorder number of vertices in $\mathcal{T}_{\mathcal{R}}(\mathcal{S})$. With all the tools that we have developed so far, we are now ready for the computation of $\mathcal{S}_r$.

Lemma 7. *$\mathcal{S}_l$ and $\mathcal{S}_r$ can be computed in $O(\log n)$ time, using a total of $O(n \log n)$ operations.*

Proof: Applying the Euler-Tour technique for tree computation on $\mathcal{T}_{\mathcal{R}}(\mathcal{S})$, we can compute the preorder number j for each vertex in $\mathcal{T}_{\mathcal{R}}(\mathcal{S})$, which determines exactly the order number $i = n - j$ of the bending step in which the corresponding rectangle is involved, during the sequential computation of $\mathcal{C}$. After that, $\mathcal{S}_r$ can be computed in $O(1)$ time, using $O(n)$ operations. ◇

Given an integer k, $2 \leq k \leq n$, instead of partitioning the (now sorted) set $\mathcal{S}$ into two subsets $\mathcal{S}_l$ and $\mathcal{S}_r$ we can partition $\mathcal{S}$ into k subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$, each containing no more than $\lceil \frac{n}{k} \rceil$ rectangles. In addition,

the boundaries of $\mathcal{S}_1, \ldots, \mathcal{S}_k$ are disjoint. $\mathcal{S}_1, \ldots, \mathcal{S}_k$ can be computed in $O(\log n)$ time, using a total of $O(n \log n)$ operations. We conclude :

Theorem 8. *A k-perfect staircase separator for a set of n axis-parallel, non-overlapping rectangles in the plane can be constructed in $O(\log n)$ time, using a total of $O(n \log n)$ operations.*

Acknowledgment

The author wishes to thank the anonymous referees for their helpful comments.

References

1. A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, and C. Yap. Parallel computational geometry. *Algorithmica*, 3:293–327, 1988.
2. M. J. Atallah and D. Z. Chen. Parallel rectilinear shortest paths with rectangular obstacles. *Comput. Geom. Theory Appl.*, 1:79–113, 1991.
3. M. J. Atallah and D. Z. Chen. On parallel rectilinear obstacle-avoiding paths. *Comput. Geom. Theory Appl.*, 3:307–313, 1993.
4. M. J. Atallah, R. Cole, and M. T. Goodrich. Cascading divide-and-conquer: A technique for designing parallel algorithms. *SIAM J. Comput.*, 18(3):499–532, 1989.
5. D. Z. Chen and K. S. Klenk. Rectilinear shortest path queries among rectangular obstacles. In *Proc. 5th Canad. Conf. Comput. Geom.*, Quebec, Canada, 1995.
6. J. Ja'Ja'. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.
7. R. E. Ladner and M. J. Fischer. Parallel prefix computation. *JACM*, 27(4):831–838, 1980.
8. P. Mitra and B. Bhattacharya. Efficient approximate shortest-path queries among isothetic rectangular obstacles. In *Proc. 3rd WADS, LNCS 709*, Springer-Verlag, pages 518–529, 1993.
9. V. H. Nguyen. Optimal binary space partitions for orthogonal objects. Diss. ETH No. 11818, Dept. of Comput. Sci., ETH Zürich, ETH-Zentrum, CH-8092 Zürich, Switzerland, 1996.
10. R. E. Tarjan and U. Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput.*, 14(4):862–874, 1985.