# Cache-Oblivious Output-Sensitive Two-Dimensional Convex Hull

Peyman Afshani[*] Arash Farzan[†]

December 16, 2007

**Abstract**

We consider the problem of two-dimensional output-sensitive convex hull in the cache-oblivious model. That is, we are interested in minimizing the number of cache faults caused when computing the convex hull of a set of $N$ points on a plane. We are interested in the output-sensitive case where number of cache misses are analyzed in the worst case based on both the input size $N$ and output size $H$ (number of extreme points that lie on the final convex hull ).

There is the lower bound of $\frac{N}{B}\log_{\frac{M}{B}}\frac{H}{B}$ to match where $M$ is the cache size and $B$ is the block size. We present a simple algorithm which almost matches this lower bound. The number of cache misses our algorithm causes is

$$O\left(\max\left\{\frac{N}{B}\log\log\frac{M}{B},\ \frac{N}{B}\log_{\frac{M}{B}}\frac{H}{B}\right\}\right)$$

. Thus, it can only be an additional term $\frac{N}{B}\log\log\frac{M}{B}$ away from the optimal.

## 1 Introduction

Efficient computation of the convex hull of a set of points on a plane is a fundamental problem in computational geometry. In the output-sensitive convex hull problem we are interested in designing an algorithm which depends both on input size (number of points) and output size (number of points on the resulting convex hull). As the number of points grow and become much larger than the cache size, the effects of cache faults start to become important in both of these problems.

Memory in today's computers consists of several layers. Algorithms were traditionally designed for a flat random access memory (RAM) with a uniform access time. However, in reality, there is an order of magnitude of difference between access times of different layers in the memory hierarchy.

The cache-aware model [1] is the simplest of hierarchical memory models which accounts for only two levels of memory. There is a cache memory of size $M$ on the first level which is divided into blocks of size $B$; on the second level there is a memory of unlimited size having the same block size.

The cache-oblivious model, introduced by Frigo *et al.* [5] is a simple and elegant multiple-level memory model and is different from the cache-aware model in that it makes no use of hardware specific parameters (such as cache or block size $M, B$). Designing algorithms without any knowledge of $M, B$ results in algorithms that behave similarly in all levels of the memory hierarchy. More specifically, if a cache-oblivious algorithm is optimal between two levels of memory hierarchy, it performs optimally on all levels of the memory hierarchy. We follow the usual practice to make the *tall cache assumption*: $M = \Omega(B^2)$.

## 2 Preliminaries and background

Given $N$ points in two dimensions there are many optimal convex hull algorithms in the RAM model that run in $O(N \log N)$. Kirkpatrick and Seidel [7] gave the first optimal output-sensitive convex hull algorithm that runs in time $O(N \log H)$ where $H$ is the size of the convex hull.

[*]Cheriton School of Computer Science, University of Waterloo, pafshani@cs.uwaterloo.ca
[†]Cheriton School of Computer Science, University of Waterloo, afarzan@cs.uwaterloo.ca

Obtaining a cache-aware or even a cache-oblivious convex hull algorithm in two dimensions is not hard. In fact, if we merely sort the point set using any cache aware/oblivious sorting algorithm [1, 5], then using Graham's scan, we can get an algorithm with total number of $O\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$ cache misses. Goodrich *et al.* [6] present an output-sensitive convex hull algorithm which makes $O\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{H}{B}\right)$ cache misses in the external-memory model. Arge and Milterson [2] showed that in the cache-aware model, these are best one can do.

We design a cache-oblivious algorithm for the output-sensitive convex hull problem. Clearly, any lower bound in the cache-aware model also holds in the cache-oblivious model.

We represent the number of points, number of points on the convex hull respectively by $N, H$ and total cache size and block size respectively by $M, B$. We also set $n = \frac{N}{B}$, $m = \frac{M}{B}$, and $h = \frac{H}{B}$.

# 3 The Algorithm

The algorithm is in fact a modification of Chan's output sensitive algorithm [3]. In this section, we consider the equivalent dual problem of finding the upper envelope of $N$ lines in plane.

First assume we are asked to output at most $H$ upper hull segments.

**Lemma 3.1.** *For a set $\mathcal{H}$ of $N$ lines in $\mathbb{R}^2$ and a parameter $H$, one can find $H$ upper hull edges in only $O(\max\{n,\ n\log_m h)\}$ cache misses.*

*Proof.* Partition $\mathcal{H}$ into $r = \frac{N}{H}$ groups of size $H$ each. For each group of $H$ lines, run the optimal (but not output sensitive) algorithm to obtain the upper hull of each group using $O(h\log_m h)$ cache misses. This will create $r$ chains such that the maximum size of each chain is $H$. We need to show that it is possible to merge these chains into a final chain causing a small number of cache misses. We need to note that this the part which deviates significantly from the original algorithm of Chan.

Let $S$ be the set of all segments constituting the chains. For a segment $s \in S$, denote the $x$-coordinate of the left end point of $s$ with $x_\ell(s)$. Partition $S$ into $h$ sets $G_1, \ldots G_H$, of size $r$ each, such that for every $i < j$ and every $u \in G_i$, $v \in G_j$ we have $x_\ell(u) < x_\ell(v)$. we use the following lemma proved by Farzan [4] (Thm. 4. 3. 1):

**Lemma 3.2.** *Given $k$ numbers in a contiguous list can be distributed into $T$ buckets $B_1, \ldots B_T$ of roughly equal size such that an element in a bucket $B_i$ is less than any element in a bucket $B_j$ with a larger index (*i.e. *$i < j$) using $O\left(\frac{k}{B}\log_m\frac{T}{B}\right)$ cache misses in the cache oblivious model.*

Hence this distribution step can be done in $O(n\log_m h)$ cache misses under the cache oblivious setting. For every $i$ the segments of $G_i$ are stored sequentially but in arbitrary order. Finally, let $x_i$ and $X_i$ denote the minimum and maximum elements of the set $\{x_\ell(u)|u \in G_i\}$ (in other words, the minimum and maximum left $x$-coordinate in $G_i$).

We merge the chains in $H$ iterations and at each iteration we process one set $G_i$, starting from $G_1$. At iteration $i+1$ we maintain the invariant that the upper hull of all the segments in $S_i = G_1 \cup G_2 \cdots \cup G_i$ to the left of the vertical line $x = X_i$ has been found. Since the rest of the segments are to the right of the vertical line $x = X_i$ this implies that upper hull of $S_i$ up to the vertical line $x = X_i$ will also appear in the upper hull of $S$. We also maintain a set $A$ of "active" segments: the set of all the segments in $S_i$ which cross the vertical line $x = X_i$. Since we have $r$ upper hull chains, the total size of set $A$ is $r$.

The above observations imply to process $G_{i+1}$ we only need to look at $A$ and $G_{i+1}$. Thus, if we do a linear search of all the segments in $A \cup G_{i+1}$ we will be able to find the next upper hull segment, if there is any. We repeat these linear search steps until we have discovered the upper hull segment at the point $x = X_{i+1}$. At this point, using one extra linear scan we update the set of active segments to contain all the segments intersecting the vertical line $x = X_{i+1}$. Each linear scan will cause $O\left(\frac{|A\cup G_{i+1}|}{B}\right) = O(\frac{r}{B})$ cache misses and thus discovery of $k_i$ upper hull edges will be accompanied by $O(\frac{rk_i}{B})$ total cache misses. At each point if we discover the $H$-th edge, we break the algorithm and output all the edges discovered by the algorithm. Since at most $H$ edges will be discovered by the convex hull algorithm, the total number of cache misses sums up to:

$$\begin{aligned} O\left(\sum_{i=1}^{H} \frac{r(k_i+1)}{B} + n\log_m h\right) &= O\left(\frac{rH}{B} + n\log_m h\right) \\ &= O(n\log_m h). \end{aligned}$$

For values $h < m$ we inflict $O(n)$ cache misses. Thus, the overall number of cache misses is $O(\max\{n,\ n\log_m h)\}$. $\square$

Finally we use the technique of guessing the output size, originally employed by Chan [3].

**Theorem 3.3.** *For a set $P$ of $N$ points in $\mathbb{R}^2$, one can compute the convex hull of $P$ with at most*

$$O\left(\max\left\{\frac{N}{B}\log\log\frac{M}{B},\ \frac{N}{B}\log_{\frac{M}{B}}\frac{H}{B}\right\}\right)$$

*cache misses in which $H$ is the size of the final convex hull.*

*Proof.* Similar to Chan's algorithm, we run the algorithm of Lemma 3.1 with values of $H$ chosen from a doubly exponential sequence. More specifically, we set $H_i = c^{2^i}$, starting from $i = 0$ until the time all the edges of the upper hull have been discovered.

For the runs of the algorithm with paramter $H$ greater than $M$, the total number of cache misses can be expressed as a geometric series:

$$\begin{aligned} \sum_{i=\log\log m}^{\log\log_c h} n\log_m c^{2^i} &= \sum_{i=\log\log m}^{\log\log_c h} 2^i n\log_m c \\ &= O(2^{\log\log_c h} n\log_m c) \\ &= O(n(\log_c h)(\log_m c)) \\ &= O(n\log_m h). \end{aligned}$$

For smaller values of the sequence, there are $O(n)$ cache misses in each run. Hence, the total number of cache misses is

$$O\left(\max\left\{n\log\log m,\ n\log_m h\right\}\right).$$

$\square$

# 4 Conclusion

We considered the problem of output-sensitive convex hull in the cache-oblivious model, and presented a simple algorithm for it. The algorithm almost matches the lower bound $O\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{H}{B}\right)$. Whether this is the optimal bound or if it can be improved is an interesting problem which we leave open.

# References


[1] A. Aggarwal and J. S. Vitter. The I/O complexity of sorting and related problems. In *ICALP '87: Proceedings of the 14th International Colloquium, on Automata, Languages and Programming*, volume 267 of *LNCS*, pages 467–478. Springer-Verlag, July 1987.

[2] L. Arge and P. B. Miltersen. On showing lower bounds for external-memory computational geometry problems. In J. Abello and J. S. Vitter, editors, *External Memory Algorithms and Visualization*, pages 139–160. American Mathematical Society Press, Providence, RI, 1999.

[3] T. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry*, 16:361–368, 1996.

[4] A. Farzan. *Cache-Oblivious Searching and Sorting in Multisets*. Master's Thesis, University of Waterloo, 2002.

[5] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 285–297. IEEE Computer Society Press, 1999.

[6] M. T. Goodrich, J.-J. Tsay, D. E. Vengroff, and J. S. Vitter. External-memory computational geometry. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 714–723, 1993.

[7] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm. *SIAM J. Comput.*, 15(1):287–299, 1986.