Computational Geometry ••• (••••) •••–•••

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Time–space trade-offs for triangulations and Voronoi diagrams

Matias Korman [a,*,1], Wolfgang Mulzer [d,1], André van Renssen [b,c], Marcel Roeloffzen [b,c], Paul Seiferth [d,1], Yannik Stein [d,1]

[a] Tohoku University, Sendai, Japan
[b] National Institute of Informatics (NII), Tokyo, Japan
[c] JST, ERATO, Kawarabayashi Large Graph Project, Japan
[d] Institut für Informatik, Freie Universität, Berlin, Germany

ARTICLE INFO

Article history:
Received 6 July 2015
Accepted 4 January 2017
Available online xxxx

Keywords:
Triangulation
Voronoi diagram
Time–space trade-off
Randomized algorithm

ABSTRACT

Let S be a planar n-point set. A *triangulation* for S is a maximal plane straight-line graph with vertex set S. The *Voronoi diagram* for S is the subdivision of the plane into cells such that all points in a cell have the same nearest neighbor in S. Classically, both structures can be computed in $O(n \log n)$ time and $O(n)$ space. We study the situation when the available workspace is limited: given a parameter $s \in \{1, \ldots, n\}$, an s-workspace algorithm has read-only access to an input array with the points from S in arbitrary order, and it may use only $O(s)$ additional words of $\Theta(\log n)$ bits for reading and writing intermediate data. The output should then be written to a write-only structure. We describe a deterministic s-workspace algorithm for computing an arbitrary triangulation of S in time $O(n^2/s + n \log n \log s)$ and a randomized s-workspace algorithm for finding the Voronoi diagram of S in expected time $O((n^2/s) \log s + n \log s \log^* s)$.

1. Introduction

Since the early days of computer science, a major concern has been to cope with strong memory constraints. This started in the '70s [22] when memory was expensive. Nowadays, a major motivation comes from a proliferation of small embedded devices where large memory is neither feasible nor desirable (e.g., due to constraints on budget, power, size, or simply to make the device less attractive to thieves).

Even when memory size is not an issue, we might want to limit the number of write operations: one can read flash memory quickly, but writing (or even reordering) data is slow and may reduce the lifetime of the storage system; write-access to removable memory may be limited for technical or security reasons (e.g., when using read-only media such as DVDs or to prevent leaking information about the algorithm). Similar problems occur when concurrent algorithms access data simultaneously. A natural way to address this is to consider algorithms that do not modify the input.

The exact setting may vary, but there is a common theme: the input resides in read-only memory, the output must be written to a write-only structure, and we can use $O(s)$ additional variables to find the solution (for a parameter s). The

* Corresponding author.
E-mail addresses: mati@dais.is.tohoku.ac.jp (M. Korman), mulzer@inf.fu-berlin.de (W. Mulzer), andre@nii.ac.jp (A. van Renssen), marcel@nii.ac.jp (M. Roeloffzen), pseiferth@inf.fu-berlin.de (P. Seiferth), yannikstein@inf.fu-berlin.de (Y. Stein).

[1] MK was supported in part by the ELC project (MEXT KAKENHI Nos. 12H00855 and 15H02665). WS and PS were supported in part by DFG Grants MU 3501/1 and MU 3501/2. YS was supported by the DFG within the research training group "Methods for Discrete Structures" (GRK 1408).

http://dx.doi.org/10.1016/j.comgeo.2017.01.001
0925-7721/

goal is to design algorithms whose running time decreases as s grows, giving a *time–space trade-off* [23]. One of the first problems considered in this model is *sorting* [19,20]. Here, the time–space product is known to be $\Omega(n^2)$ [8], and matching upper bounds for the case $b \in \Omega(\log n) \cap O(n/\log n)$ were obtained by Pagter and Rauhe [21] (b denotes the available workspace in *bits*).

Our current notion of memory constrained algorithms was introduced to computational geometry by Asano et al. [4], who showed how to compute many classic geometric structures with $O(1)$ workspace (related models were studied before [9]). Later, time–space trade-offs were given for problems on simple polygons, e.g., shortest paths [1], visibility [6], or the convex hull of the vertices [5].

We consider a model in which the set S of n points is in an array such that random access to each input point is possible, but we may not change or even reorder the input. Additionally, we have $O(s)$ variables (for a parameter $s \in \{1, \ldots, n\}$). We assume that each variable or pointer contains a data word of $\Theta(\log n)$ bits. Other than this, the model allows the usual word RAM operations. In this setting we study two problems: computing an arbitrary triangulation for S and computing the Voronoi diagram $\mathrm{VD}(S)$ for S. Since the output cannot be stored explicitly, the goal is to report the edges of the triangulation or the vertices of $\mathrm{VD}(S)$ successively, in no particular order. Dually, the latter goal may be phrased in terms of Delaunay triangulations. We focus on Voronoi diagrams, as they lead to a more natural presentation.

Both problems can be solved in $O(n^2)$ time with $O(1)$ workspace [4] or in $O(n \log n)$ time with $O(n)$ workspace [7]. However, to the best of our knowledge, no trade-offs were known before. Our triangulation algorithm achieves a running time of $O(n^2/s + n \log n \log s)$ using $O(s)$ variables. A key ingredient is the recent time–space trade-off by Asano and Kirkpatrick for triangulating a special type of simple polygons [3]. This also lets us obtain significantly better running times for the case that the input is sorted in x-order; see Section 2. For Voronoi diagrams, we use random sampling to find the result in expected time $O((n^2 \log s)/s + n \log s \log^* s))$; see Section 3. Together with recent work of Har-Peled [16], this appears to be one of the first uses of random sampling to obtain space–time trade-offs for geometric algorithms. The sorting lower bounds also apply to triangulations and Voronoi diagrams (since we can reduce the former to the latter). This implies that our second algorithm is almost optimal.

2. A time–space trade-off for general triangulations

In this section we describe an algorithm that outputs the edges of a triangulation for a given point set S in arbitrary order. For ease in the presentation we first assume that S is presented in sorted order. In this case, a time–space trade-off follows quite readily from known results. We then show how to generalize this for arbitrary inputs, which requires a careful adaptation of the existing data structures.

2.1. Sorted input

Suppose the input points $S = \{q_1, \ldots, q_n\}$ are stored in increasing order of x-coordinate and that all x-coordinates are distinct, i.e., $x_i < x_{i+1}$ for $1 \le i < n$, where x_i denotes the x-coordinate of q_i.

A crucial ingredient in our algorithm is a recent result by Asano and Kirkpatrick for triangulating *monotone mountains*[2] (or *mountains* for short). A mountain is a simple polygon with vertex sequence $v_1, v_2, \ldots, v_k$ such that the x-coordinates of the vertices increase monotonically. The edge $v_1 v_k$ is called the *base*. Mountains can be triangulated very efficiently with bounded workspace.

Theorem 2.1 *(Lemma 3 in [3], rephrased). Let H be a mountain with n vertices, stored in sorted x-order in read-only memory. Let $s \in \{2, \ldots, n\}$. We can report the edges of a triangulation of H in $O(n \log_s n)$ time and using $O(s)$ words of space.*

Since S is given in x-order, the edges $q_i q_{i+1}$, for $1 \le i < n$, form a monotone simple polygonal chain. Let $\mathrm{Part}(S)$ be the subdivision obtained by the union of this chain with the edges of the convex hull of S (denoted by $\mathrm{conv}(S)$). A convex hull edge is *long* if the difference between its indices is at least two (i.e., the endpoints are not consecutive). The following lemma (illustrated in Fig. 1) lets us decompose the problem into smaller pieces.

Lemma 2.2. *Any bounded face of* $\mathrm{Part}(S)$ *is a mountain whose base is a long convex hull edge. Moreover, no point of S lies in more than four faces of* $\mathrm{Part}(S)$.

Proof. Any point $q_i \in S$ has at most four neighbors in $\mathrm{Part}(S)$: q_{i-1}, q_{i+1}, its predecessor and its successor along the convex hull (if q_i lies on $\mathrm{conv}(S)$). Thus, no point of S belongs to more than four faces of $\mathrm{Part}(S)$.

Next we show that every face F of $\mathrm{Part}(S)$ is a mountain with a *long* convex-hull edge as its base. The boundary of F contains at least one long convex-hull edge $e = (q_i, q_j)$ $(i < j)$, as other edges connect only consecutive vertices. Since the monotone path $q_i, \ldots, q_j$ forms a cycle with the edge e and since the boundary of F is a simple polygon, we conclude that

[2] Also known as *unimonotone polygons* [15].

Please cite this article in press as: M. Korman et al., Time–space trade-offs for triangulations and Voronoi diagrams, Comput. Geom. (2017), http://dx.doi.org/10.1016/j.comgeo.2017.01.001