Delaunay Triangulations in $O(\text{sort}(n))$ and Other Transdichotomous and Hereditary Algorithms in Computational Geometry

Kevin Buchin Wolfgang Mulzer

Everyone knows that sorting n numbers takes $\Omega(n \log n)$ time—and yet this bound can often be beaten. With van Emde Boas (vEB) trees, we can sort n items from a universe U in $O(n \log \log |U|)$ time on a pointer machine. For *transdichotomous* models (ie, a word RAM), Fredman and Willard introduced *fusion trees*, which triggered off a development that culminated in the $O(n\sqrt{\log \log n})$ sorting algorithm by Han and Thorup. In computational geometry, many results use vEB trees or similar structures to surpass traditional lower bounds. However, all these results are *orthogonal*: they assume that the input is rectilinear or can be efficiently approximated by a rectilinear structure. In 1992, Willard studied fusion trees in the context of computational geometry, and he asked how quickly Delaunay triangulations (DTs) can be computed on a word RAM. The breakthrough came in 2006/07, when Chan and Pătraşcu discovered transdichotomous algorithms for point location in nonorthogonal planar subdivisions, which led to better bounds for many classical problems. For planar DTs and convex hulls in $\mathbb{R}^3$, they achieved a rather unusual running time of $n2^{O(\sqrt{\log \log n})}$, which raises the question whether their result is optimal. More generally, Chan and Pătraşcu asked if the approach via point location is inherent, and they also briefly discussed the various approaches to transdichotomous algorithms and why the fusion method seemed most feasible for point location. In particular, it remained open whether the vEB approach can provide any benefits for computational geometry apart from the orthogonal results mentioned above.

The fusion method makes strong use of the word RAM—the main idea is to "fuse" (parts of) several data items into one word and then use constant time bit-operations for parallel processing. Since fusion-tree-based results need all the bit-fiddling power, they usually do not generalize to other computational models. By contrast, the vEB method relies on hashing: data is organized as a high-degree tree, and the higher-order bits of a data item are used to locate the appropriate child at each step. Thus, it applies in any model where the hashing step can be done in constant time, eg, a pointer machine. This is useful for *hereditary* results, where we need to preprocess a large universe U for queries about subsets of U. Here, vEB trees show that sorting a big set once suffices to sort any given—large enough—subset of U faster than in $\Theta(n \log n)$ time.

Our results. We start with a randomized reduction from nearest-neighbor graphs (NNGs) to Delaunay triangulations (DTs). Our method uses a new variant of randomized incremental constructions (RICs) that employ dependent sampling for faster conflict location. The running time of our reduction is proportional to the time needed to find the NNG plus the expected number of tetrahedra a standard RIC creates, which is often linear (for example if the point set is planar). Our algorithm is quite simple and works in any dimension, but the analysis turns out to be rather subtle. The main idea is to compute the DT of a sample and to then locate the remaining points

by tracing along the edges of the NNG. However, the NNG is usually not connected, and the DT of the sample might not contain a starting point for every component. To get around this, we must bias the sample such that it meets every component of the NNG. The main challenge is to show that the dependent choices in the sampling procedure do not have any negative effect on the running time.

It is a well-known fact that the NNG of a point set can be computed in linear time once we know a quadtree for it. Hence: *Given a quadtree for a point set $P \subseteq \mathbb{R}^d$, we can compute the Delaunay triangulation of P,* $\mathrm{DT}(P)$*, in time proportional to the expected structural change of a RIC.* Thus, even though DTs appear to be inherently non-orthogonal, we actually only need the information encoded in quadtrees, a highly orthogonal structure. Many results follow. First, we answer Willard's seventeen-year-old question by showing that planar DTs—and hence Voronoi diagrams and related structures like Euclidean minimum spanning trees—can be computed in time $O(\mathtt{sort}(n))$ on a word RAM.[1] Since our reduction works in a traditional model, we also get pointer machine algorithms for hereditary DTs: we can preprocess a planar point set U such that for any subset $P \subseteq U$, it takes $O(|P| \log\log |U|)$ time to find $\mathrm{DT}(P)$. Since a planar quadtree can be computed by an algebraic decision tree (ADT) of linear depth once the points are sorted according to the x- and y-direction, we find that after presorting in *two* orthogonal directions, a planar DT can be computed by an ADT of expected linear depth. This is particularly curious, since there is an $\Omega(n \log n)$ lower bound when the points are sorted in *one* direction, and also for 3D convex hulls when the points are sorted in any constant number of directions. This problem has appeared in the literature for at least twenty years, and our result seems to mark the first non-trivial progress on this question. However, we do not know if a quadtree for presorted points can actually be constructed in linear time, since the algorithms we know still need an $\Omega(n \log n)$ overhead for data handling. All the above results generalize to higher dimensions, but there we also need to account for the structural change of the RIC. The reduction also implies traditional lower bounds for computing well-separated pair decompositions and nearest-neighbor graphs if the points are presorted in one direction, since otherwise Delaunay triangulations could be computed faster in this case.

In the second part, we extend the result about hereditary DTs to 3-polytopes and describe a vEB-like data structure for this problem: preprocess a point set $U \subseteq \mathbb{R}^3$ in general convex position such that the convex hull of any $P \subseteq U$ can be found in time $O(|P|(\log\log |U|)^2)$. To achieve this, we use a relatively recent technique which we call *scaffolding*: in order to find many related structures quickly, we first compute a "typical" instance—the *scaffold* S—in a preprocessing phase. To answer a query, we insert the input points into S and use a fast *hereditary* algorithm to remove the scaffold. We also need a careful decomposition of the universe and a bootstrapping method similar to the one by Chan and Pătraşcu, continuing their main theme that transdichotomous geometric algorithms require a careful analysis of the structure of geometric objects. To the best of our knowledge, this is the first non-orthogonal vEB-style data structure in computational geometry. Also, this improves a recent algorithm from SoCG 2009 for splitting a 3-polytope whose vertices are colored with $\chi \geq 2$ colors into its monochromatic parts. All our algorithms are randomized, and it is an interesting open problem to derandomize them. In particular, it would be very interesting to find a deterministic algorithm for splitting DTs or, more generally, convex polytopes.

[1] We require one non-standard, but AC^0, operation, the *shuffle*. However, we believe that it should be straightforward to adapt existing integer sorting algorithms so that our result also holds on a more standard word RAM, and in our paper we demonstrate this for the comparatively simple sorting algorithm by Andersson et al.