Theoretical Computer Science 1055 (2025) 115491

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Deterministic protocols for Voronoi diagrams and triangulations of planar point sets on the congested clique ☆

Jesper Jansson [a], Christos Levcopoulos [b], Andrzej Lingas [b,*], Valentin Polishchuk [c], Quan Xue [d]

[a] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[b] Department of Computer Science, Lund University, Lund, Sweden
[c] Communications and Transport Systems, ITN, Linköping University, Sweden
[d] The University of Hong Kong, Hong Kong, China

ARTICLE INFO

Section Editor: Paul G. Spirakis

Handling Editor: Yong Chen

Keywords:
Voronoi diagram
Delaunay triangulation
Convex hull
Distributed algorithm
The congested clique model

ABSTRACT

We study the problems of computing the Voronoi diagram and a triangulation of a set of n^2 points with $O(\log n)$-bit coordinates in the Euclidean plane in a substantially sublinear in n number of rounds in the congested clique model with n nodes. First, we observe that if the points are uniformly at random distributed in a unit square then their Voronoi diagram within the square can be computed in $O(1)$ rounds with high probability (w.h.p.). Next, we show that if a very weak smoothness condition is satisfied by an input set of n^2 points with $O(\log n)$-bit coordinates in the unit square then the Voronoi diagram of the point set within the unit square can be deterministically computed in $O(\log n)$ rounds in this model. Finally, we present a deterministic $O(\log n)$-round protocol for a triangulation of n^2 points with $O(\log n)$-bit coordinates in the Euclidean plane. It relies on our novel method for extending triangulations of two planar point sets separated by a straight line to a complete triangulation of the union of the sets in $O(1)$ rounds.

1. Introduction

The *congested clique* is a relatively new model of communication and computation introduced by Lotker et al. in 2005 [9]. It focuses on the cost of communication between the nodes in a network, ignoring the cost of local computation within each node. Hence, it can be seen as opposite to the Parallel Random Access Machine (PRAM) model, studied extensively in the 80s and 90s. The PRAM model focuses on the computation cost and ignores the communication cost [1].

Originally, the complexity of many dense graph problems has been studied in the congested clique model under the following assumptions. Each node of the congested clique represents a distinct vertex of the input graph and knows that vertex's neighborhood in the graph. Every node also knows the unique ID numbers (between 1 and n) of itself and all the other nodes at the start of the

☆ This article is an extended version of the paper entitled "The Voronoi Diagram of Weakly Smooth Planar Point Sets in $O(\log n)$ Deterministic Rounds on the Congested Clique" that has been presented at the 30th International Computing and Combinatorics Conference (COCOON 2024). It also includes among other things a fragment of the paper entitled "Convex Hulls and Triangulations of Planar Point Sets on the Congested Clique" presented at the Thirty-Fifth Canadian Conference on Computational Geometry (CCCG 2023).

* Corresponding author.
E-mail addresses: jj@i.kyoto-u.ac.jp (J. Jansson), Christos.Levcopoulos@cs.lth.se (C. Levcopoulos), Andrzej.Lingas@cs.lth.se (A. Lingas), valentin.polishchuk@alumni.stonybrook.edu (V. Polishchuk), csxuequan@connect.hku.hk (Q. Xue).

https://doi.org/10.1016/j.tcs.2025.115491
Received 28 November 2024; Received in revised form 28 June 2025; Accepted 19 July 2025

Available online 28 July 2025

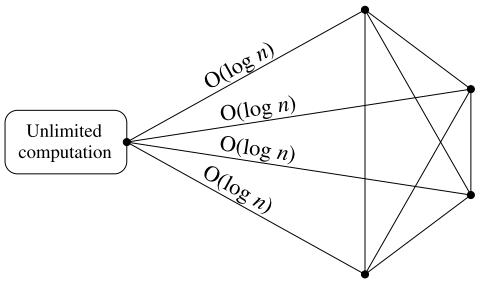

Fig. 1. Illustrating the congested clique model. In every round, each node can send a distinct message of $O(\log n)$ bits to each other node.

computation. The computation proceeds in rounds. In every round, each of the n nodes can send a distinct message of $O(\log n)$ bits to each other node and also perform unlimited local computation; see Fig. 1. The primary complexity objective is to minimize the number of rounds necessary to solve a given problem on the input graph in this model.

For several basic graph problems such as the minimum spanning tree problem, one has even succeeded to design $O(1)$-round protocols in the congested clique model [10,13] (a protocol is another name for a distributed algorithm). Observe that when the input graph is of bounded degree and the edge weights have $O(\log n)$-bit representations, each node can just send the ID numbers of all nodes in its neighborhood and the weights of its incident edges to, e.g., the first node, in $O(1)$ rounds. After that, the first node can solve the whole problem locally. However, such a trivial solution would require $\Omega(n)$ rounds when the input graph is dense.

Matrix problems [3], sorting and routing [7], and geometric problems [6] have also been studied in the congested clique model. In all cases, the basic input items, i.e., matrix entries or keys, or points in the plane, respectively, are assumed to have $O(\log n)$-bit representations and each node initially has a batch of n such items. Note that the bound on bit representation of an input item is a natural consequence of the $O(\log n)$-bit bound on the size of a single message which makes input items of unbounded bit representation incompatible with the assumed model. As in the graph case, in every round, each node can send a distinct $O(\log n)$-bit message to each other node and perform unlimited local computation. Significantly, it has been shown that matrix multiplication can be performed in a number of rounds substantially sublinear in n [3] while sorting and routing can be implemented in $O(1)$ rounds (Theorems 4.5 and 3.7 in [7]).

Already at the end of the 90s, Goodrich [5] presented $O(1)$-round randomized protocols for the construction of the three-dimensional convex hull of a set of points in three-dimensional Euclidean space in $O(1)$ communication rounds in the so-called Bulk Synchronous Processing model (BSP). (BSP is similar to PRAM but it takes into account the communication and synchronization costs in contrast to PRAM.) His result also implies an $O(1)$-round bound on the randomized construction of the Voronoi diagram and the dual Delaunay triangulation of a planar point set in the BSP model. By using the $O(1)$-round routing protocol of Lenzen [7] (see also Fact 1), Goodrich's $O(1)$ bound on the number of rounds necessary for the construction of the Voronoi diagram and Delaunay triangulation can be carried over from the BSP model to ours. In [5], Goodrich also presented a deterministic $O(1)$-round protocol for the two-dimensional convex hull in the BSP model that, again by applying the protocol from [7], works in $O(1)$-rounds on the congested clique with n nodes.

In this context, the major open problems are to derive non-trivial upper bounds on the number of rounds sufficient to deterministically construct: (i) the convex hull of a point set in the three-dimensional Euclidean space, and, in particular (ii) the Voronoi diagram or the dual Delaunay triangulation of a planar point set, and even just (iii) a triangulation of a planar point set.

The bottleneck in the design of efficient parallel or distributed algorithms for the Voronoi diagram of a planar point set using a direct divide-and-conquer approach is an efficient parallel or distributed merging of Voronoi diagrams. Aggarwal et al. [1] presented an $O(\log^2 n)$-time CREW PRAM algorithm for the Voronoi diagram based on an involved $O(\log n)$-time PRAM method for the parallel merging. (In the CREW variant of PRAM, multiple processors can read a memory cell but only one can write at a time.) Subsequently, Amato and Preparata [2] demonstrated an $O(\log n)$-time CREW PRAM algorithm for the three-dimensional convex hull and consequently also for the two-dimensional Voronoi diagram of a point set.

Our new results are as follows. To begin with, we observe that if the points are uniformly at random distributed in a unit square then their Voronoi diagram within the square can be computed in $O(1)$ rounds w.h.p.

Next, we substantially extend the local approach to the construction of the Voronoi diagram used in the design of parallel and distributed algorithms for the Voronoi diagram of points drawn uniformly at random, e.g., from a unit square [8,14]. We show that already a very weak smoothness condition on the input set of n^2 points with $O(\log n)$-bit coordinates within a unit square is sufficient to obtain an $O(\log n)$ upper bound on the number of rounds required to construct the Voronoi diagram of the set within the unit square on the congested n-clique. Roughly, our weak smoothness condition says that if a square Q of side length ℓ within the unit square contains at least n out of the n^2 input points then any square of the same size at distance at most $4\sqrt{2}\ell$ from Q and within the unit square has to contain at least one input point. We obtain our result by combining a quadtree partition of the unit square with a local construction of the Voronoi diagram within the squares corresponding to the leaves of quadtree. The local construction is possible due to the fulfillment of the weak smoothness condition.

Finally, we present a deterministic $O(\log n)$-round protocol for a triangulation of n^2 points with $O(\log n)$-bit coordinates in the Euclidean plane. It is based on our novel method for completing triangulations of two planar point sets separated by a straight line to a full triangulation of the union of the sets in $O(1)$ rounds.

Table 1
Deterministic protocols for the construction of the Voronoi diagram or a triangulation of the input set of n^2 points with $O(\log n)$-bit coordinates in the Euclidean two-dimensional space E^2 presented in this article.

input set of n^2 points in E^2	problem	# rounds	theorem
uniform random in unit square	Voronoi diagram	$O(1)$	Theorem 1
$(\frac{1}{2}, 4\sqrt{2})$-smooth	Voronoi diagram	$O(\log n)$	Theorem 3
arbitrary	triangulation	$O(\log n)$	Theorem 4

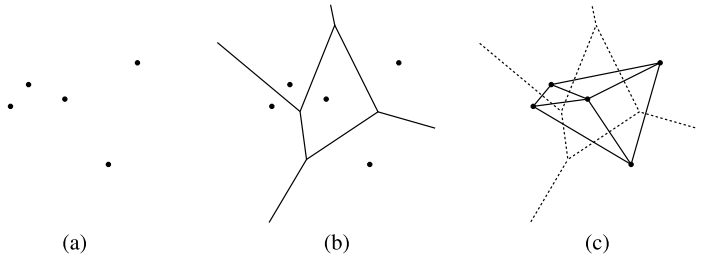

Fig. 2. An example of a planar point set, its Voronoi diagram, and the dual Delaunay triangulation.

We summarize our results in Table 1.

In order to simplify the presentation, we assume throughout the article that the points in the input point sets are in general position (i.e., neither any three input points are co-linear nor any four input points are co-circular). We also refer to the points of the input point set as *vertices*, while reserving the word *nodes* to refer to the communicating parties in the underlying congested clique network.

Our article is structured as follows. The next section contains basic mathematical/geometric definitions and facts on routing and sorting in the congested clique model. Section 3 shows our $O(1)$-round protocol for the Voronoi diagram of a uniform random planar point set in a unit square. Section 4 presents our $O(\log n)$-round protocol for the Voronoi diagram and Delaunay triangulation of a weakly smooth planar point set within a square. Our $O(\log n)$-round protocol for a triangulation of a planar point set is presented in Section 5. We conclude with final remarks in Section 6.

2. Preliminaries

The cardinality of a set S is denoted by $|S|$. For a positive integer r, $[r]$ stands for the set of positive integers not exceeding r.

For a finite set S of points in the Euclidean plane, the *Voronoi diagram of* S is the partition of the plane into $|S|$ regions such that each region consists of all points in the plane having the same closest point in S; see Fig. 2.

A *Delaunay triangulation of* S is a maximal set of non-crossing edges between pairs of points from S such that no point from S is placed inside any of the formed triangles' circumcircles. It is well known that if no four points in S are co-circular then the Delaunay triangulation of S is a dual of the Voronoi diagram of S in the following sense [12]: for each edge e of each region in the Voronoi diagram of S, if e is a part of the bisector of the points u, v in S then (u, v) is an edge of the Delaunay triangulation of S; again, see Fig. 2.

Let $S = \{p_1, ..., p_n\}$ be a set of n distinct points in the Euclidean plane such that the x-coordinate of each point is not smaller than that of p_1 and not greater than that of p_n. The *convex hull of* S is the smallest convex polygon P for which every $q \in S$ lies in the interior of P or on the boundary of P. The *upper hull* of S (with respect to (p_1, p_n)) is the part of the convex hull of S beginning in p_1 and ending in p_n in clockwise order. Symmetrically, the *lower hull* of S (with respect to (p_1, p_n)) is the part of the convex hull of S beginning in p_n and ending in p_1 in clockwise order.

A *supporting line* for the convex hull or upper hull or lower hull of a finite point set in the Euclidean plane is a straight line that touches the hull without crossing it properly. Let S_1, S_2 be two finite sets of points in the Euclidean plane separated by a vertical line. The *bridge* between the upper (or lower) hull of S_1 and the upper (or, lower, respectively) hull of S_2 is a straight line that is a supporting line for both of the upper (lower, respectively) hulls. See Fig. 3 for an illustration.

Our concept of weak smoothness is formally defined in terms of two parameters as follows.

Definition 1. Let ε, d be two positive real constants. A set of N points in a unit square is (ε, d)-*smooth* if for any two equal size, axis parallel squares Q, R within the unit square the following implication holds:

if Q contains at least N^{ε} points of S and R is at distance at most $d \cdot \ell$ from Q, where ℓ is the length of each edge of Q and R, then R contains at least one point of S.

Intuitively, the smoothness means that if a square Q contains sufficiently many points, then there must be a point nearby in the square R.

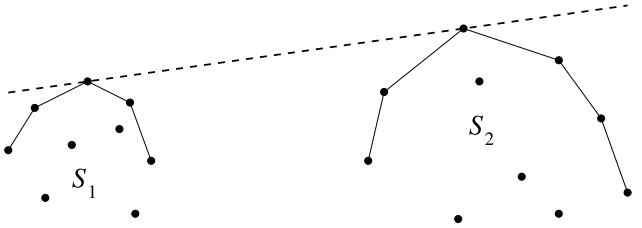

Fig. 3. An example of the bridge between the upper hulls of S_1 and S_2.

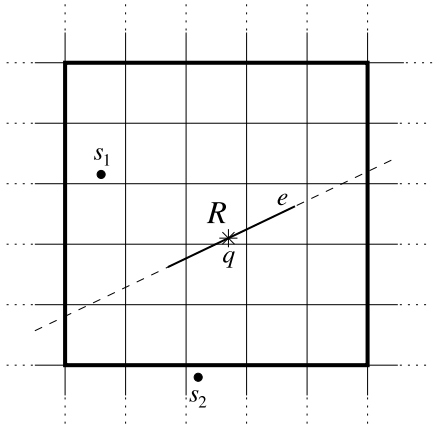

Fig. 4. An example of the configuration in the proof of Lemma 1. The basic square R is the small square in the middle of the figure, e is an edge of the Voronoi diagram that passes through R, and $TL_i(R)$ consists of the 25 basic squares inside the big square indicated by thick lines.

We also need to define a sequence of grids within a unit square and related notions.

Definition 2. For a nonnegative integer i, we shall denote by $G_i(U)$ the orthogonal grid within the unit orthogonal square U that includes the edges of U such that the distance between two neighboring vertical or horizontal grid line segments is $\frac{1}{2^i}$. A *basic square* of $G_i(U)$ is a square within U such that the endpoints of each of its edges are a pair of neighboring grid points. For a basic square R in $G_i(U)$, we shall denote the orthogonal region consisting of R and the two layers of basic squares around R by $TL_i(R)$ (if there are less than two layers between R and an edge of the unit square then $TL_i(R)$ includes only zero or one layers from that side).

The proof of the following lemma corresponds to the second paragraph of the proof of Theorem 4 in [6].

Lemma 1. *Let R be a basic square in a grid $G_i(U)$ within the unit square U. Consider a finite set S of points within the unit square. If R contains a point in S then the Voronoi diagram of S within R can be computed by taking into account only the points of S within $TL_i(R)$. Hence, in particular all edges (u, v) of the Delaunay triangulation of S such that a part of the bisector of u and v borders some region of the Voronoi diagram of S within R can be determined.*

Proof. Let e be an edge of the Voronoi diagram of S within R. The edge e has to be a part of the bisector of some pair of points s_1 and s_2 in S. Consider an arbitrary point q on e. Suppose that s_1 or s_2 is placed outside $TL_i(R)$, i.e., the orthogonal area consisting of at most $1 + 8 + 16 = 25$ squares including R. See Fig. 4. Without loss of generality, let s_2 be such a point. Then the distance between q and s_2 is at least $2 \cdot \frac{1}{2^i}$, while the distance between q and every point inside R is at most $\sqrt{2} \cdot \frac{1}{2^i}$. We obtain a contradiction because R contains at least one point from S and q is closer to such a point than to s_2. □

Lenzen gave an efficient solution to the following fundamental routing problem in the congested clique model, known as the *Information Distribution Task* (IDT) [7]:
Each node of the congested n-clique holds a set of exactly n $O(\log n)$-bit messages with their destinations, with multiple messages from the same source node to the same destination node allowed. Initially, the destination of each message is known only to its source node. Each node is the destination of exactly n of the aforementioned messages. The messages are globally lexicographically ordered by their source node, their destination, and their number within the source node. For simplicity, each such message explicitly contains these values, in particular making them distinguishable. The goal is to deliver all messages to their destinations, minimizing the total number of rounds.

Lenzen proved that by partitioning the nodes into subsets and applying a bipartite multigraph edge-coloring algorithm to rearrange the messages before sending them to their final destinations, IDT can be solved in 16 rounds (Theorem 3.7 in [7]). He also noted that the relaxed IDT, where each node is required to send and receive at most n messages, easily reduces to IDT and is therefore also solvable in $O(1)$ rounds. From here on, we shall refer to this important result as:

Fact 1. [7] The relaxed Information Distribution Task can be solved deterministically within $O(1)$ rounds.

The *Sorting Problem* (SP) is defined as follows:
Each node i of the congested n-clique holds a set of n $O(\log n)$-bit keys. All the keys are different w.l.o.g. Each node i needs to learn all the keys of indices in $[n(i-1)+1, ni]$ (if any) in the total order of all keys.

Lenzen showed that SP can be solved in 37 rounds if each node holds a set of exactly n keys (Theorem 4.5 in [7]). In order to relax the requirement that each node holds exactly n keys to that of with at most n keys, we can determine the maximum key and add appropriate different dummy keys in $O(1)$ rounds. We summarize this result as follows:

Fact 2. [7] The relaxed Sorting Problem can be solved in $O(1)$ rounds.

3. Voronoi diagram of uniform random planar point sets

When the n^2 points with $O(\log n)$-bit coordinates are drawn uniformly at random from a unit square or circle then the expected number of required rounds to compute the Voronoi diagram or the dual Delaunay triangulation on the congested clique becomes $O(1)$ (cf. [8,14]). This can be deduced from Goodrich's $O(1)$-round randomized protocol for the Voronoi diagram of arbitrary planar point sets in the BSP model [5]. Simply, while running Goodrich's protocol on the uniform random point set, one can use the input points' coordinates as the source of randomness for the protocol. However, Goodrich's general protocol is quite involved and many details are missing in the available proceedings version. In this section, we demonstrate that Lemma 1 combined with the Chernoff bounds yields a much simpler protocol for the Voronoi diagram or Delaunay triangulation for a planar point set drawn uniformly at random from a unit square.

We need to recall the Chernoff bounds and the union bound.

Fact 3. (multiplicative Chernoff lower bound) Suppose $X_1, ..., X_n$ are independent random variables taking values in $\{0,1\}$. Let X denote their sum and let $\mu = E[X]$ denote the sum's expected value. Then, for any $\delta \in [0,1]$, $Prob(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}}$ holds. Similarly, for any $\delta \geq 0$, $Prob(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2+\delta}}$ holds.

Fact 4. (The union bound) For a sequence $A_1, A_2, ..., A_r$ of events,
$Prob(A_1 \cup A_2 \cup ... \cup A_r) \leq \sum_{i=1}^{r} Prob(A_i)$.

We shall say that an event dependent on n^2 input points in the plane holds with high probability (w.h.p.) if its probability is at least $1 - \frac{1}{n^\alpha}$ asymptotically (i.e., there is an integer n_0 such that for all $n \geq n_0$, the probability is at least $1 - \frac{1}{n^\alpha}$), where α is a constant not less than 2.

Theorem 1. *The Voronoi diagram of n^2 points with $O(\log n)$-bit coordinates drawn uniformly at random from a unit square in the Euclidean plane can be computed within the square w.h.p. in $O(1)$ rounds on the congested clique.*

Proof. Let S denote the set of the n^2 input points. Partition the unit square into n squares of size $\frac{1}{\sqrt{n}} \times \frac{1}{\sqrt{n}}$, i.e., consider the n basic squares in the grid $G_{\frac{1}{2}\log n}(U)$. For any of the n squares, the expected number of points from S that lie inside it is n, and so by using $\mu = n$ and selecting $\delta = \sqrt{\frac{7 \ln n}{n}}$ in Fact 3, we see that each of the n squares contains $\Theta(n)$ points from S with probability at least $1 - \frac{1}{n^3}$.

Assign to each of the n squares R a distinct clique node. Deliver to each node assigned R all the points in $TL_{\frac{1}{2}\log n}(R)$ in $O(1)$ rounds w.h.p. by Lenzen's routing protocol given in Fact 1. The total number of points that need to be delivered to each node is $O(n)$ w.h.p. since each of the at most $1+8+16=25$ basic squares in $TL_{\frac{1}{2}\log n}(R)$ contains $O(n)$ points w.h.p.

For the delivery purpose, each node locally computes the grid $G_{\frac{1}{2}\log n}(U)$ and the assignment of the basic squares in the grid to clique nodes in the same way. This enables each node to figure out the node destinations of the points held by it. Note that if a point p belongs to a basic square Q in $TL_{\frac{1}{2}\log n}(R)$, where R is some basic square in $G_{\frac{1}{2}\log n}(U)$, then symmetrically, R has to be included in $TL_{\frac{1}{2}\log n}(Q)$. Since each node holds $O(n)$ points w.h.p. and each point p has to be sent solely to the at most 25 nodes assigned to the basic squares in $TL_{\frac{1}{2}\log n}(Q)$, where Q is the basic square containing p, all the points can be sent and delivered to their destinations by running Lenzen's protocol $O(1)$ times, totally in $O(1)$ rounds.

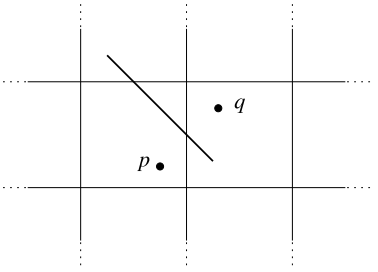

Fig. 5. An example of a Voronoi edge stretching through three basic squares in $G_{\frac{1}{2}\log n}(U)$.

By Lemma 1, each node can compute the Voronoi diagram within the square R it is assigned to locally with probability at least $1-\frac{1}{n^3}$. For this purpose, any standard sequential algorithm for the Voronoi diagram of a planar point set can be used, e.g., [4]. Hence, the union bound (Fact 4) implies that the probability that all nodes compute their local Voronoi diagrams correctly is at least $1-\frac{1}{n^2}$.

It remains to output the union of the Voronoi diagrams. Local Voronoi edges within neighboring squares belonging to the bisector of the same pair of input points have to be glued together; see Fig. 5. For this purpose, each node equips each Voronoi edge e in its square with a prefix (p,q), where p, q are the input points on whose bisector e is placed. Then, all the local Voronoi edges are sorted by their prefixes in $O(1)$ rounds by employing Fact 2. Assuming that each square contains $\Theta(n)$ points, for each given prefix there are $O(1)$ local Voronoi edges that need to be glued.

Simply, if a local edge with a prefix (p,q) occurs in a basic square R in $G_{\frac{1}{2}\log n}(U)$ then p and q have to belong to $TL_{\frac{1}{2}\log n}(R)$ by Lemma 1, since R contains $\Theta(n)$ input points. E.g., consider the basic square Q in $TL_{\frac{1}{2}\log n}(R)$ that contains the point q. Then, symmetrically R has to be included in $TL_{\frac{1}{2}\log n}(Q)$. We conclude that there are at most 25 such squares R in $G_{\frac{1}{2}\log n}(U)$, and consequently there are at most 25 local edges sharing the prefix (p,q).

After the sorting of the local edges by prefixes, each node can easily glue the complete families of at most 25 local edges sharing the same prefix. It can happen that two consecutively numbered clique nodes share such a single family. Then, they can exchange $O(1)$ messages in order to compute the whole edge gluing the local edges from the family. □

4. The extended local approach

Consider a $(\frac{1}{2},4\sqrt{2})$-smooth set of n^2 points with $O(\log n)$-bit coordinates in a unit orthogonal square. We shall first describe a protocol for listing the edges of the Delaunay triangulation of the set that are dual to the edges of the Voronoi diagram of the set within the unit square. Roughly, it implicitly grows a quadtree of squares rooted at the unit square in phases corresponding to the levels of the quadtree. If a square R currently at a leaf of the quadtree and the two layers of equal size squares around it jointly include at most cn input points, for an appropriate positive constant c, then the intersection of the Voronoi diagram of the input point set with R and the dual edges of the Delaunay triangulation of the input point set can be computed locally. This follows from the smoothness condition and Lemma 1 combined with the fact that the parent square of R and the two layers of equal size squares around it jointly include more than cn input points. Otherwise, four child squares whose union forms R are created on the next level of the quadtree. In particular, checking the aforementioned condition in parallel for the squares at the current front level of the quadtree and delivering the necessary points to the nodes representing respective frontier squares in $O(1)$ rounds on the congested n-clique are highly non-trivial.

protocol $DT-SQUARE(S,U)$
Input: A $(\frac{1}{2},4\sqrt{2})$-smooth set of n^2 points with $O(\log n)$-bit coordinates in a unit orthogonal square U held in n-point batches at the n nodes of the congested clique.
Output: The set of the edges of the Delaunay triangulation of S dual to the edges of the Voronoi diagram of S within U held in $O(n)$-edge batches at clique nodes.

1. Initialize an empty list L of edges of the Delaunay triangulation of S.
2. Activate the basic square U in $G_0(U)$ and assign it to the first node.
3. For $i=0,1,\ldots$ do
   (a) Each node for each point p in its batch determines the number $num(p)$ of the basic square of $G_i(U)$ containing p in a common fixed numbering of the basic squares in $G_i(U)$ (e.g., column-wise and top-down-wise in each column). Next, a *prefixed representation* of p is formed by the concatenation of the fixed size bit representation of $num(p)$ with the fixed size bit representations of the coordinates of p.
   (b) The points in S are sorted by their prefixed representation. After that each node informs all other nodes about the range of numbers of the basic squares in $G_i(U)$ holding the prefixed representations of points in S that landed at the node after the sorting of the prefixed representations of all the points.

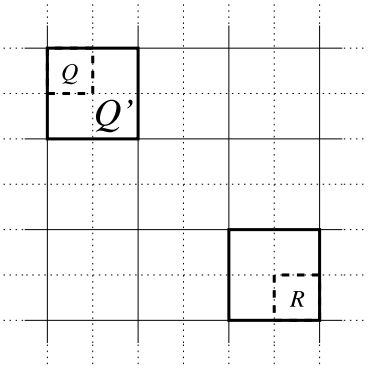

Fig. 6. An example of the configuration in the proof of Lemma 3.

(c) For each basic square W in $G_i(U)$ such that the prefixed representations of points belonging to W that are held in a sequence C of at least two consecutive nodes, the nodes in C inform additionally the other nodes in C about the number of the prefixed representations of the points in W they got so in particular the node in C with the smallest index can compute the total number of the points in W.
(d) For each active basic square R in $G_i(U)$, the node representing R sends queries to the nodes holding the prefixed point representations of the points in the basic squares in $TL_i(R)$ (i.e., in R and the two layers of basic squares around R in $G_i(U)$) about the number of points in these squares. If several nodes hold the prefixed point representation of points in a basic square in $TL_i(R)$, the query is sent only to that node with the smallest index.
(e) After getting answers to the queries, each node proceeds as follows for each active basic square in R in $G_i(U)$ it represents. If the total number of points of S in $TL_i(R)$ does not exceed $100n$ then the node asks the nodes holding the prefixed representations of the points in the basic squares in $TL_i(R)$ for sending the points to the node. After that the node computes the Voronoi diagram of all these points and then the intersection of the diagram with R locally. Next, the node appends to L all edges (u, v) where a part of the bisector of u and v borders some region of the Voronoi diagram in the computed intersection. Otherwise, the node activates the four basic squares in $G_{i+1}(U)$ whose union forms R and assigns them temporarily to itself.
(f) The nodes balance the assignment of active basic squares in $G_{i+1}(U)$ by informing all other nodes about the number of active basic squares in $G_{i+1}(U)$ they are assigned and following the results of the same assignment balancing algorithm run by each of them separately locally.
(g) The list L is sorted in order to remove multiple copies of the same edge.

Lemma 2. *$DT-SQUARE(S,U)$ activates basic squares solely in the grids $G_i(U)$, where $i = O(\log n)$.*

Proof. Simply, the points in S have $O(\log n)$-bit coordinates so at depth at most $O(\log n)$ the condition in Step 3(e) of $DT-SQUARE(S,U)$ has to be satisfied. □

Lemma 3. *The protocol $DT-SQUARE(S,U)$ is correct.*

Proof. When the Voronoi diagram of the points of S in $TL_i(R)$ for a basic square R of the grid $G_i(U)$ is computed then there must be square Q' in the grid $G_{i-1}(U)$ that contains at least $100n/25$ points in S and is at distance at most $\frac{\sqrt{2}}{2^{i-1}}$ from the basic square in $G_{i-1}(U)$ that is the parent of R. Hence, there is a basic square Q in $G_i(U)$ that is part of Q' and contains at least $100n/100$ points in S; see Fig. 6. By straightforward calculations, the distance between Q and R is at most $4\sqrt{2}\frac{1}{2^i}$. Thus, by the assumed $(\frac{1}{2}, 4\sqrt{2})$-smoothness property, the square R contains at least one point in S. It follows from Lemma 1 that the intersection of the Voronoi diagram of the points of S in $TL_i(R)$ with R yields the Voronoi diagram of S within R. Hence, the edges appended to the list L are the edges of the Delaunay triangulation of S dual to the edges of the Voronoi diagram of S within U. It easily follows by induction on i during forming the quadtree of active basic squares that the leaf active basic squares form a partition of the unit square U. Therefore, for each edge (u, v) of the Delaunay triangulation of S dual to an edge of the Voronoi diagram of S within U there must exist a positive integer i and an active basic square R in $G_i(U)$ such that R does not have any child active basic squares in $G_{i+1}(U)$ and a part of the bisector of u and v borders some region in the Voronoi diagram of S within R. Hence, the list L is complete. □

Lemma 4. *For $i = 0, 1, \ldots, O(\log n)$, the number of active basic squares in the grid $G_i(U)$ is $O(n)$ during the execution of $DT-SQUARE(S,U)$.*

Proof. We argue similarly as at the beginning of the proof of Lemma 3. If R is an active basic square in $G_i(U)$ different from the unit square U then there must exist a basic square Q in $TL_{i-1}(R')$, where R' is the parent of R in $G_{i-1}(U)$, such that Q contains at

least $100n/25$ points in S. Now it is sufficient to note that: (i) there are at most $O(n)$ basic squares in $G_{i-1}(U)$ that contain at least $100n/25$ points in S; (ii) there are at most $O(1)$ basic squares Q' in $G_{i-1}(U)$ different from R' such that Q is included in $TL_{i-1}(Q')$; (iii) an active basic square in $G_{i-1}(U)$ can be a parent to at most four active basic squares in $G_i(U)$. □

Lemma 5. *The protocol $DT-SQUARE(S,U)$ can be implemented in $O(\log n)$ rounds on the congested clique.*

Proof. Steps 1, 2 can be easily implemented in $O(1)$ rounds. By Lemma 2, the block under the for loop in Step 3 is iterated $O(\log n)$ times. It is sufficient to show that this block (a-g) can be implemented in $O(1)$ rounds.

Step 3(a) can be performed totally locally.

The sorting of the prefixed representations of points in S in Step 3(b) can be done in $O(1)$ rounds by Fact 2.

For each node, the range of the numbers of the basic squares in $G_i(U)$ holding the prefixed representations of points in S at the node after the sorting of the prefix representations of the points can be specified by two $O(\log n)$-bit numbers. Hence, all nodes can inform all other nodes about their ranges in $O(1)$ rounds. Thus, Step 3(b) requires $O(1)$ rounds in total.

The situation described in Step 3(c) can happen for at most n basic squares W in $G_i(U)$. It requires sending by each node at most two different messages to at most n nodes in total and also receiving at most n messages. Hence, Step 3(c) can be implemented in $O(1)$ rounds by using the routing protocol from Fact 1.

In Step 3(d), for each active basic square, a node representing the square has to send $O(1)$ $O(\log n)$-bit queries to $O(1)$ other nodes. The total number of active basic squares in $G_i(U)$ is $O(n)$ by Lemma 4. Hence, by using the routing protocol from Fact 1 this task can be done in $O(1)$ rounds.

Consider Step 3(e). Answering the queries sent in Step 3(c) can be done by local computations and the routing reverse to that in Step 3(d) in $O(1)$ rounds. After that, each node for each active square assigned to it determines locally if the criterion for computing the Voronoi diagram of S within R is satisfied. If so the node sends messages asking the nodes holding the prefixed representations of points in the squares of $TL_i(R)$ for sending the points. This requires sending $O(n)$ messages for each active basic square in $G_i(U)$. Since the total number of such squares is $O(n)$ by Lemma 4 and each node represents $O(1)$ active squares in $G_i(U)$, it can be accomplished in $O(1)$ rounds by Fact 1. Delivering the requested points to the nodes representing respective active basic squares can also be done in $O(1)$ rounds for the following reasons. For each active basic square the node representing it needs to receive $O(n)$ points. Furthermore, by Lemma 4 there are $O(n)$ active basic squares in $G_i(U)$. Hence, since the active squares are assigned to the n nodes in a balanced way, each node needs to receive $O(n)$ points. Also, the points contained in a given basic square in $G_i(U)$ can be requested by at most $O(1)$ nodes since there are at most $O(1)$ active basic squares behind these requests to the given square. Since the sorted prefixed representations of the points in S are divided between the nodes in a balanced way, each node needs to send $O(n)$ points, each of them to $O(1)$ nodes. We conclude that this part of Step 3(d) can be implemented in $O(1)$ rounds by Fact 1. The remaining parts of Step 3(d) are done locally.

Step 3(f) requires sending and receiving by each node $O(1)$ messages so it can be done in $O(1)$ rounds.

Consider an edge (u,v) dual to some edge of the Voronoi diagram of the points of S included in $TL_i(R)$ within an active basic square R in $G_i(U)$. The edge can be appended to L at most for $O(1)$ different squares R as u, v are in $TL_i(R)$. Therefore, the list L may contain at most $O(1)$ copies of an edge of the Delaunay triangulation of S so Step 3(g) can be implemented in $O(1)$ rounds by using the sorting protocol from Fact 2. □

Lemmata 3 and 5 yield the first main result of this section.

Theorem 2. *Let S be a $(\frac{1}{2}, 4\sqrt{2})$-smooth set of n^2 points with $O(\log n)$-bit coordinates in an orthogonal unit square, held in n-point batches at the n nodes of the congested clique. The set of edges of the Delaunay triangulation of S dual to the edges of the Voronoi diagram of S within the unit square can be constructed in $O(\log n)$ rounds on the congested clique.*

Lemma 6. *Let S be defined as in Theorem 2. Suppose that a list L of the edges of the Delaunay triangulation of S dual to the edges of the Voronoi diagram of S within the unit square is held in $O(n)$-edge batches at the n nodes of the congested clique. The Voronoi diagram of S within the unit square can be constructed in $O(1)$ rounds on the congested clique.*

Proof. Double the list L by inserting for each $(u,v) \in L$ also (v,u) into L. For each edge (u,v) locally determine an $O(\log n)$-bit representation of the angle $\beta(u,v)$ between (u,v) and the horizontal line passing through u. For instance, the representation can specify the tangent of the angle by $(v_y - u_y, v_x - u_x)$. Sort the edges (x,y) by $(x, \beta(u,v))$, letting the nodes compare the angle tangents locally, using the sorting protocol from Fact 2. In this way, for each point $u \in S$, a sub-list of all edges of the Delaunay triangulation incident to u in the angular order is created. Some of the sub-lists can stretch through several nodes of the clique network. Given the edges of the Delaunay triangulation incident to u in the angular order, the edges of the Voronoi region of u within the unit square can be easily produced. This is done by intersecting the bisectors of u and the other endpoints of consecutive edges incident to u in the angular order as long as the intersection of two consecutive bisectors is within the unit square. Otherwise, the border of the region of u has to be filled with the fragment of the perimeter of the unit square between the intersections of the two bisectors with the perimeter. □

Theorem 2 combined with Lemma 6 yields the second main result of this section.

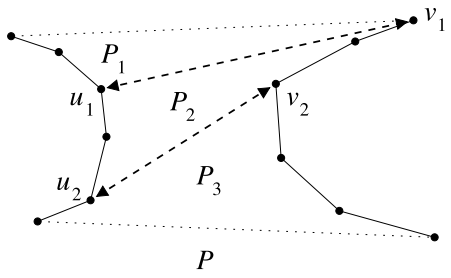

Fig. 7. An example of the partition of the polygon P into the subpolygons $P_1, \ldots, P_{k+1}$ in the procedure $Triangulate$.

Theorem 3. *Let S be a $(\frac{1}{2}, 4\sqrt{2})$-smooth set of n^2 points with $O(\log n)$-bit coordinates in an orthogonal unit square held in n-point batches at the n nodes of the congested clique. The Voronoi diagram of S within the unit square can be constructed in $O(\log n)$ rounds on the congested clique.*

5. Planar point set triangulation in $O(\log n)$ rounds

Our method of triangulating a set of n^2 points in the congested n-clique model initially resembles the merging method of constructing the convex hull of the points [6]. That is, first the input point set is sorted by x-coordinates. Then, each node triangulates its sorted batch of n points locally. Next, the triangulations are pairwise merged and extended to triangulations of doubled point sets by using the procedure $Merge$ in parallel in $O(\log n)$ phases. In the general case, the procedure $Merge$ calls the procedure $Triangulate$ in order to triangulate the area between the sides of the convex hulls of the two input triangulations, facing each other.

The main idea of the procedure $Triangulate$ is to select up to $n-2$ inner vertices v with almost equal interdistances on the longer of the convex hulls sides and send their coordinates and the coordinates of their neighbors to the nodes holding the facing side of the other hull (in particular, if the longest chain contains at most n vertices then all its inner vertices are selected). The latter nodes send back candidates (if any) for highest mates u of the selected vertices v (i.e., highest vertices u on the facing side of the other hull such that the segments between v and u can be edges of a triangulation extending the existing partial triangulation). The segments are used to split the polygon area to triangulate into several subpolygons that are triangulated by several recursive calls of $Triangulate$ in parallel. See Fig. 7. Before the recursive calls, the edges bordering the subpolygons are moved to new node destinations so edges of each of the subpolygons are kept in a sequence of consecutive clique nodes. This is done by a global routing in $O(1)$ rounds serving all parallel calls of $Triangulate$ on a given recursion level, for a given phase of $Merge$ (its first argument).

Since the recursion depth of $Triangulate$ is $O(1)$ and $Merge$ is run in $O(\log n)$ phases, the total number of required rounds becomes $O(\log n)$.

To simplify the presentation, we shall assume that the size n of the clique network is a power of 2.

procedure $Triangulation(S)$

1. Sort the points in S by their x-coordinates so each node receives a subsequence consisting of n consecutive points in S, in the sorted order.
2. Each node sends the first point and the last point in its subsequence to the other nodes.
3. Each node q constructs a triangulation $T(q,q)$ of the points in its sorted subsequence locally.
4. For $1 \le p < q \le n$, $T(p,q)$ will denote a triangulation of the points in the sorted subsequence held in the nodes p through q. For $i = 0, \ldots, \log n - 1$, in parallel, for $j = 1, 1+2^{i+1}, 1+2\cdot 2^{i+1}, 1+3\cdot 2^{i+1}, \ldots$ the union of the triangulations $T(j, j+2^i-1)$ and $T(j+2^i, j+2^{i+1}-1)$ is transformed to a triangulation $T(j, j+2^{i+1}-1)$ of the sorted subsequence held in the nodes j through $j+2^{i+1}-1$ by calling the procedure $Merge(i,j)$.

procedure $Merge(i,j)$
Input: A triangulation $T(j, j+2^i-1)$ of the subsequence held in the nodes j through $j+2^i-1$ and a triangulation $T(j+2^i, j+2^{i+1}-1)$ of the subsequence held in the nodes $j+2^i$ through $j+2^{i+1}-1$.
Output: A triangulation $T(j, j+2^{i+1}-1)$ of the subsequence held in the nodes j through $j+2^{j+1}-1$.

1. Compute the bridges between the convex hulls of $T(j, j+2^i-1)$ and $T(j+2^i, j+2^{i+1}-1)$.
2. Determine the polygon P formed by the bridges between the convex hulls of $T(j, j+2^i-1)$ and $T(j+2^i, j+2^{i+1}-1)$, the right side of the convex hull of $T(j, j+2^i-1)$, and the left side of the convex hull of $T(j+2^i, j+2^{i+1}-1)$ between the bridges.
3. $Triangulate(P, j, j+2^{i+1}-1)$

procedure $Triangulate(P,p,q)$
Input: A simple polygon P composed of two convex chains facing each other on opposite sides of a vertical line and two edges crossing the line, held in nodes p through q, with $p \le q$.
Output: A triangulation of P held in nodes p through q.

1. If $p = q$ then the p node triangulates P locally and terminates the call of the procedure.
2. The nodes p through q determine the lengths of the convex chains on the border of P and set ℓ to the length of the longest chain. Then, the nodes holding the longest chain (in case of ties, the left chain) select $k = \min\{n-2, \ell-1\}$ inner vertices on the longest chain with almost equal interdistances. Next, these nodes send the coordinates of all the selected vertices and all vertices adjacent to the selected ones on the chain to all the nodes p through q holding the vertices of the opposite chain.
3. Each node holding some vertices of the opposite chain, for each received, selected vertex v, determines the highest vertex u on the opposite chain held by the node such that (v,u) forms a diagonal of P (if any). (The node can verify if (v,u) is a diagonal of P by checking if the segment (v,u) is within the intersection of the union of the half-planes on the side of P induced by the edges adjacent to v with the union of the half-planes on the side of P induced by the edges adjacent to u.) If (v,u) is well defined then the node sends it as a candidate diagonal (v,u) to the node holding v.
4. Each node holding the vertices of the longest convex chain, for each selected vertex v, picks the diagonal (v,u) with highest u among the received candidate diagonals as the diagonal to draw from v. Then, it sends the coordinates of (v,u) to all other nodes p through q.
5. The nodes p through q split the polygon P into subpolygons $P_1, \ldots, P_{k+1}$ by the picked diagonals (v,u). Next, they compute the new destinations for the edges of the subpolygons $P_1, \ldots, P_{k+1}$ so P_i can be held in nodes r_{i-1} through r_i, where $r_0 = p$, $r_{i-1} \leq r_i$, $r_{k+1} = q$, for $i = 1, \ldots, k+1$.
6. A synchronized global routing in $O(1)$ rounds corresponding to the current phase of the calls to the procedure $Merge$ (given by its first argument) and all parallel calls of the procedure $Triangulate$ on the same recursion level is implemented by using Fact 1. In particular, the edges of P_1 through P_{k+1} are moved to the new consecutive destinations among nodes p through q.
7. In parallel, $Triangulate(P_i, r_{i-1}, r_i)$ are performed for $i = 1, \ldots, k+1$.

The correctness of the procedure $Triangulate$ follows from the following lemma.

Lemma 7. *Let P be an input polygon for the procedure $Triangulate$. Next, let v be an inner vertex on a convex chain of P and let u be the highest vertex on the opposite convex chain of P such that (v,u) is a diagonal of P. Similarly, let v' be another inner vertex on the convex chain of P that v is located below v and let u' be the highest vertex on the opposite convex chain of P such that (v',u') is a diagonal of P. The diagonals (v,u) and (v',u') do not intersect, i.e., u' cannot be placed over u on the opposite chain.*

Proof. The proof is by contradiction. Suppose that the two diagonals (v,u) and (v',u') properly intersect. Then, (v,u') is also a diagonal of P and u' is higher than u, a contradiction with the definition of u. □

At the beginning, we have outlined our triangulation method, in particular the procedures forming it, in a top-down fashion. We now complement this outline with a bottom-up analysis.

In the analysis of the procedure $Triangulate(P,p,q)$, it is important to remember that several calls of this procedure can be run on the congested clique in parallel. The parallel calls of $Triangulate$ occur on the same recursion level, for a given phase of the parallel calls of procedure $Merge(i,\ )$, i.e., for given i. Here, it is also important to note that a single node of the congested clique can be involved in at most two such non-local calls shared with the preceding node or the following node (i.e., of the form $Triangulate(P,p,q)$, where $p < q$). Besides this, the node can be assigned several calls of the procedure on smaller subpolygons totally held within the node (i.e., of the form $Triangulate(P,p,p)$) that the node can solve locally. In particular, Steps 2-4, 6 can involve sending and/or receiving $O(n)$ messages for a single node in the implementation of the parallel calls of the procedure which can be achieved in $O(1)$ rounds by using Lenzen's global routing protocol. To implement the remaining steps but for the recursive calls, it is sufficient to use the nodes p through q in $O(1)$ rounds. The recursion depth of $Triangulate$ is $O(1)$. Simply, $O(1)$ recursion depth is enough to reduce the lengths of the convex chains in the resulting subpolygons to at most $n-1$. After that, diagonals are drawn from all inner vertices of the longest chain in a subpolygon so further $O(1)$ recursion levels are sufficient to obtain a complete triangulation. We conclude that the parallel calls of this procedure can be implemented in $O(1)$ rounds.

The first two steps of the procedure $Merge(i,j)$, i.e., constructing the bridges and then the polygon P between the convex hulls, can be implemented in $O(1)$ rounds by using the $O(1)$-round convex hull algorithm from [5] on the nodes j through $j + 2^{i+1} - 1$ (by [7], our application of the convex hull algorithm requires $O(1)$ rounds on the congested $\sqrt{n2^{i+1}}$-clique and can be easily implemented in $O(1)$ rounds by the $2^i + 1$ nodes of the congested n-clique). Finally, the call to $Triangulate$ in the last step of $Merge$ requires $O(1)$ rounds by our analysis of $Triangulate$. We conclude that $Merge(i,j)$ can be implemented in $O(1)$ rounds.

Finally, all steps in $Triangulation(S)$ except the one involving parallel calls to $Merge(i,j)$ in $O(\log n)$ phases can be done in $O(1)$ rounds. For a given phase, i.e., given i, each node is involved in $O(1)$ parallel calls of $Merge(i,j)$, not counting the auxiliary global routing steps. It follows from our analysis of $Merge(i,j)$ and $i = O(\log n)$ that $Triangulation(S)$ can be implemented in $O(\log n)$ rounds.

Theorem 4. *Consider a congested n-clique network, where each node holds a batch of n points in the Euclidean plane specified by $O(\log n)$-bit coordinates. A triangulation of the set S of the n^2 input points can be computed by the procedure $Triangulation(S)$ in $O(\log n)$ rounds on the congested clique.*

6. Final remarks

The *message complexity* of a protocol in the congested clique model is the maximum total number of $O(\log n)$-bit messages exchanged by the n nodes of the congested clique during a run of the protocol (e.g., see [11]). In case of our protocols, it is easily seen to be the product of the maximum number of messages that can be exchanged in a single round, i.e., $\Theta(n^2)$, times the number of required rounds. Thus, the message complexity of our deterministic protocols for the Delaunay triangulation and the Voronoi diagram of n^2 point sets from Section 4 and the message complexity of our deterministic triangulation protocol for the input point set from Section 5 is $O(n^2 \log n)$ while that of our protocol for uniform random planar point sets from Section 3 is only $O(n^2)$.

The remaining major open problem is the derivation of a low polylogarithmic upper bound on the number of rounds sufficient to deterministically construct the Voronoi diagram of n^2 points with $O(\log n)$-bit coordinates in the Euclidean plane (when the points are not necessarily randomly distributed) on the congested clique with n nodes. This seems feasible but it might require a substantial effort as in the PRAM case [1,2]. Even the design of a faster, i.e., sub-logarithmic, deterministic protocol for the construction of a triangulation of n^2 points with $O(\log n)$-bit coordinates in the Euclidean plane seems to be an interesting open problem.

CRediT authorship contribution statement

Jesper Jansson: Writing – review & editing, Writing – original draft, Visualization, Validation, Investigation, Formal analysis, Conceptualization. **Christos Levcopoulos:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization. **Andrzej Lingas:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization. **Valentin Polishchuk:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization. **Quan Xue:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors are grateful to the anonymous reviewers for valuable comments. This research was in part funded by Swedish Research Council grant 2018-04001 and JSPS KAKENHI JP20H05964.

References

[1] A. Aggarwal, B. Chazelle, L.J. Guibas, C. Ó'Dúnlaing, C.-K. Yap, Parallel computational geometry, Algorithmica 3 (1988) 293–327.
[2] M. Amato, F.P. Preparata, A time-optimal parallel algorithm for three-dimensional convex hull, Algorithmica 14 (2) (1995) 169–182.
[3] K. Censor-Hillel, P. Kaski, J.H. Korhonen, C. Lenzen, A. Paz, J. Suomela, Algebraic methods in the congested clique, Distrib. Comput. 32 (6) (2019) 461–478.
[4] S. Fortune, A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (1987) 153–174.
[5] M.T. Goodrich, Randomized fully-scalable BSP techniques for multi-searching and convex hull construction, in: Proceedings of the Eighth Annual Symposium on Discrete Algorithms (SODA 1997), ACM-SIAM, 1997, pp. 767–776.
[6] J. Jansson, C. Levcopoulos, A. Lingas, V. Polishchuk, Convex hulls, triangulations, and Voronoi diagrams of planar point sets on the congested clique, arXiv: 2305.09987, 2023, Preliminary version in: Proceedings of the Thirty-Fifth Canadian Conference on Computational Geometry (CCCG 2023), 2023, pp. 183–189.
[7] C. Lenzen, Optimal deterministic routing and sorting on the congested clique, in: Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing (PODC 2013), ACM, 2013, pp. 42–50.
[8] C. Levcopoulos, J. Katajainen, A. Lingas, An optimal expected-time parallel algorithm for Voronoi diagrams, in: Proceedings of the First Scandinavian Workshop on Algorithm Theory (SWAT 88), in: Lecture Notes in Computer Science, vol. 318, Springer-Verlag, 1988, pp. 190–198.
[9] Z. Lotker, B. Patt-Shamir, E. Pavlov, D. Peleg, Minimum-weight spanning tree construction in $O(\log \log n)$ communication rounds, SIAM J. Comput. 35 (1) (2005) 120–131.
[10] K. Nowicki, A deterministic algorithm for the MST problem in constant rounds of congested clique, in: Proceedings of the Fifty-Third Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021), ACM, 2021, pp. 1154–1165.
[11] S. Pemmaraju, V. Sardeshmukh, Super-fast MST algorithms in the congested clique using $o(m)$ messages, in: Proceedings of the 36th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016), LIPICS, 2016, pp. 47:1–47:15.
[12] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Texts and Monographs in Computer Science., vol. 10, Springer-Verlag, 1985.
[13] P. Robinson, Brief announcement: what can we compute in a single round of the congested clique?, in: Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing (PODC 2023), ACM, 2023, pp. 168–171.
[14] B.C. Vemuri, R. Varadarajan, N. Mayya, An efficient expected time parallel algorithm for Voronoi construction, in: Proceedings of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1992), ACM, 1992, pp. 392–401.