A Tight Bound for Facial Distance Patterns in Planar Graphs

Viktor Fredslund-Hansen Shay Mozes Oren Weimann

Abstract

Let G be an undirected unweighted planar graph and let $S = (s_0, \ldots, s_{k-1})$ be the vertices of a designated face, listed in cyclic order. Consider a vector that stores the distances from an arbitrary vertex v to all vertices of S. The *pattern* of v is obtained by taking the difference between every pair of consecutive values in this vector. Li and Parter [STOC'19] proved an upper bound of $O(k^3)$ on the number of unique patterns over all vertices of G. We improve this to $O(k^2)$, matching a known lower bound and settling a conjecture in [ISAAC'22]. The simple proof was found by OpenAI's GPT 5.6-Sol model.

Plugging this new bound into known results has the following three immediate implications for undirected unweighted planar graphs: (1) it gives an improved compression of the Okamura–Seymour metric, (2) it improves the space required by constant-time exact distance oracles, and (3) it improves the fastest distributed algorithm for computing the diameter. We further present a previously unknown and nontrivial implication: a (centralized) $\tilde{O}(n^{8/5})$-time algorithm for computing the diameter, improving over the $\tilde{O}(n^{5/3})$ algorithm of [SODA'18] which works for weighted directed planar graphs. Thus, there is currently a gap between the time for computing the diameter between weighted and unweighted planar graphs.

1 Introduction

Let G be an undirected unweighted planar graph and let $S = (s_0, \ldots, s_{k-1})$ be the vertices of a designated face, listed in cyclic order. The *pattern* p^v of a vertex v in G is the vector $p^v = \langle d_G(v, s_1) - d_G(v, s_0), d_G(v, s_2) - d_G(v, s_1), \ldots, d_G(v, s_{k-1}) - d_G(v, s_{k-2})\rangle$ where $d_G(x, y)$ denotes the x-to-y distance in G. Let $p_\#$ denote the number of unique patterns over all vertices of G. Li and Parter [11] proved that $p_\# = O(k^3)$, and used this bound to obtain an $\tilde{O}(D^5)$-round algorithm for computing the diamater in the CONGEST model of distributed computation. Their bound was later used by Fredslund-Hansen, Mozes, and Wulff-Nilsen [6] to construct exact distance oracles with constant query time and subquadratic space for unweighted undirected planar graphs. Mozes, Wallheimer, and Weimann [12] gave a simple $\Omega(k^2)$ lower bound and conjectured that the upper bound should be $O(k^2)$. This problem and conjecture was recently mentioned by Da Wei Zheng as an important open problem in a Dastuhl seminar on Metric Sketching and Dynamic Algorithms for Geometric and Topological Graphs [2, Open problem 5.6]. We provide a simple proof of this conjecture, which we found by a single prompt to OpenAI's GPT 5.6-Sol model. It is surprising (not to say embarrasing) that this open problem has such a simple proof, which has eluded the community despite the human efforts invested in it.

The Li-Parter proof. Since the graph is unweighted and undirected, every entry of p^v is in $\{-1, 0, 1\}$ by the triangle inequality. The patterns can be transformed[1] to be binary (i.e. over $\{-1, 1\}$ instead of $\{-1, 0, 1\}$) vectors of dimension $2k - 1$ by subdividing every edge of

[1] This transformation was presented in [12] and in the conference talk of [11].

the graph. Li and Parter's proof is based on the simple observation that there cannot be two vertices v and u and 4 indices $a < b < c < d$ such that $p^u_a = -1, p^u_b = 1, p^u_c = -1, p^u_d = 1$ but $p^v_a = 1, p^v_b = -1, p^v_c = 1, p^v_d = -1$. The reason is that such forbidden $(-1, 1, -1, 1), (1, -1, 1, -1)$ induced patterns correspond to an illegal configuration of shortest paths in planar graphs.

Now, consider the patterns of all vertices of G as the rows of a binary matrix P. The *VC-dimension* of P is the largest number d such that there exists in P a submatrix of d columns that contains all possible 2^d different rows. The above forbidden configuration implies that the VC-dimension of P is at most 3. By Sauer's Lemma [14], this means that there are $O((2k-1)^3) = O(k^3)$ distinct rows. This is their entire proof.

Limitations of the Li-Parter proof. The following set of sequences over $\{-1, 1\}^{k-1}$ was observed by [12]: $\left\{(-1)^{x_1} \circ 1^{x_2} \circ (-1)^{x_3} \circ 1^{k-1-x_1-x_2-x_3} \mid x_1 + x_2 + x_3 < k\right\}$. There is no pair of sequences in this set that contains the forbidden $(1, -1, 1, -1), (-1, 1, -1, 1)$ configuration, and yet its cardinality is $\Theta(k^3)$. This shows that VC-dimension alone is not enough to break the $O(k^3)$ upper bound.

The new proof. We prompted ChatGPT with a description of the state of the art and asked for any improvement on the upper bound. It replied that the answer is $p_\# = O(k^2)$. Specifically, it noted that since the sequence of vertices along the distinguished face is cyclic, the entries of the pattern (as a vector in $\{-1, +1\}^{2k-1}$) must sum up to either 1 or -1 because if we added one more term that closes the cycle back to s_0, the sum would telescope to 0. It follows that the number of 1's in any pattern is either k or $k-1$. This property was previously observed, but has not been exploited. It then pointed out that sets of vectors in $\{-1, +1\}^{2k-1}$ that avoid the forbidden configuration and have the same number of 1's are in fact what is known as *weakly separated subsets of* $[2k-1]$[2]. The notion of weakly separated subsets was defined and studied by Leclerc and Zelevinsky [10] in the context of quasi-commuting quantum flag minors. They proved (Theorem 1.2 in [10]) that the maximal size of a family of weakly separated subsets of $[n]$ is $\frac{1}{2}(n+2)(n+1)+1$. Their inductive proof is elementary and takes about one page. Using their result in our case (i.e. applying their bound with $[n] = [2k-1]$ twice; once for the patterns with k 1's and once for the patterns with $k-1$ 1's) immediately gives an upper bound of $4k^2 + k + 2 = O(k^2)$ on $p_\#$. Leclerc and Zelevinsky also proved a tight bound of $k(n-k)+1$ for weakly separated k-subsets of $[n]$[3]. The proof of this bound in [10] is longer and more complicated. Using this bound in our case yields a slightly better upper bound of $2k^2 - 2k + 2$. In addition, ChatGPT gave an elegant and simple proof that directly gives the stronger bound on the size of a set of vectors in $\{-1, +1\}^n$ that exclude the forbidden configuration, without going through the stronger definition of weakly separated subsets of $[n]$[4]. We clarified, further simplified, and cleaned that proof and present it in Section 2.

Implications. There are several implications of this new $p_\# = O(k^2)$ bound:

- **Okamura–Seymour metric compression [13].** This problem asks to compactly encode the S-to-T distances between the vartices of a face S and an arbitrary subset of vertices T of an undirected unweighted planar graph G. A query (v, s_i) to the encoding (with $v \in T$ and $s_i \in S$) returns the v-to-s_i distance in G. Li and Parter [11] (see also comment in [12])

[2]Here, and throughout we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Each vector corresponds to the set of coordinates of its 1 entries.

[3]Substitute $\ell = k$ into the statement of Theorem 1.3 in [10].

[4]Without cardinality constraints, two weakly separated subsets correspond to pattens that avoid the forbidden configuration, but not the other way around.

presented a compression of size $\tilde{O}(k \cdot p_{\#} + |T|)$ and $O(1)$ query time. This was improved by Mozes, Wallheimer and Weimann [12] to $\tilde{O}(p_{\#} + |T|)$ space with $\tilde{O}(1)$ query time. By plugging $p_{\#} = O(k^3)$ their bound was $\tilde{O}(k^3 + |T|)$. Plugging in the new $O(k^2)$ bound immediately improves this to $\tilde{O}(k^2 + |T|)$.

- **Distance oracles with constant query time [6].** This problem asks for a compact data structure that reports exact distances in G (between any pair of vertices) in *constant* time. Fredslund-Hansen, Mozes and Wulff-Nilsen [6] showed how to use patterns to obtain such an oracle for undirected unweighted planar graphs with $O(n^{7/4})$ space. By plugging in $p_{\#} = O(k^2)$ their space immediately improves to $O(n^{5/3})$.
- **Distributed diameter computation [11].** This problem is in distributed computing, in the CONGEST model. It asks for the smallest number of rounds required to compute the diameter D of an undirected unweighted planar graph. Li and Parter [11] used patterns to obtain an algorithm requiring $\tilde{O}(D^5)$ rounds. Mozes, Wallheimer and Weimann [12] observed that the same algorithm can be easily tweaked to run in $\tilde{O}(D^4)$ rounds. By plugging in $p_{\#} = O(k^2)$ this immediately improves to $\tilde{O}(D^3)$ rounds.
- **Centralized diameter computation**. The current fastest algorithm to compute the diameter of a planar graph (by Gawrychowski, Kaplan, Mozes, Sharir and Weimann [7]) runs in $\tilde{O}(n^{5/3})$ time and works even for weighted directed planar graphs. Improving its running time is a major open problem in planar graph algorithms. In Section 3 we present a non-trivial use of the new $p_{\#} = O(k^2)$ bound to achieve a faster $\tilde{O}(n^{8/5})$ time algorithm for diameter in unweighted undirected planar graphs. Thus currently there is gap between weighted and unweighted diameter computation times.

2 The ChatGPT Proof

In this section we prove the main theorem:

Theorem 1. *Let G be a connected, undirected, unweighted planar graph with a designated face f. Let k be the number of edges of the boundary walk of f. Then $p_{\#} \in O(k^2)$.*

We first state and prove Lemma 1, which bounds the maximum cardinality of a family of vectors in $\{-1, +1\}^n$ with exactly r 1's. We then provide, for the sake of completeness, the already known arguments that imply the main theorem.

Lemma 1. *Let $0 \le r \le n$. The maximal size of a family $\mathcal{F}$ of vectors in $\{-1, +1\}^n$, each with exacty r 1's, that avoids the forbidden configuration is $r(n - r) + 1$.*

Proof. Assume that $1 \le r \le n - 1$, as otherwise the claim is trivial.

Step 1: encode the family by a prefix trie. Let T be the rooted prefix trie whose nodes are all prefixes of the vectors in $\mathcal{F}$ (the root corresponds to the empty prefix), with children given by appending 1 or -1 whenever the extended prefix occurs. All vectors have length n, so the leaves of T are exactly the vectors of $\mathcal{F}$, and the number of leaves is $L = |\mathcal{F}|$. Call an internal node *branching* if it has both children, and let b and u be the numbers of branching and single-child internal nodes. The tree has $u + b + L$ nodes, hence $u + b + L - 1$ edges, and counting edges by out-degree gives $u + 2b$. Therefore

$$L = b + 1, \tag{1}$$

and it suffices to prove $b \le r(n-r)$.

Step 2: map branching nodes into a rectangle. For a binary prefix P set $x(P) = \#\{1\text{'s in } P\}$ and $y(P) = \#\{\text{-1's in } P\}$. If P is branching, its -1-child extends to a full vector with exactly $n-r$ -1's, so $y(P)+1 \le n-r$; its 1-child extends to a vector with exactly r 1's, so $x(P)+1 \le r$. Hence every branching node satisfies

$$0 \le y(P) \le n-r-1, \qquad 0 \le x(P) \le r-1,$$

and the location map $\lambda(P) = (x(P), y(P))$ sends branching nodes into a rectangle of exactly $r(n-r)$ lattice points (see Fig. 1).

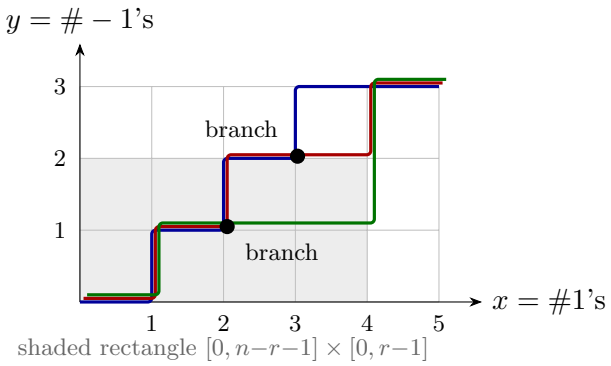

Figure 1: A family $\mathcal{F}$ of vectors with $n = 8$ and $r = 5$ as monotone lattice paths (1 = right, -1 = up), sharing prefixes as in the trie; slight offsets distinguish coinciding segments. Branching prefixes (dots) are confined to the shaded $(n-r) \times r$ rectangle, and the proof of Lemma 1 shows that no two branching prefixes occupy the same lattice point; hence $b \le r(n-r)$ and $|\mathcal{F}| = b+1$.

Step 3: λ is injective on branching nodes. Suppose two distinct branching nodes $P \ne Q$ satisfy $\lambda(P) = \lambda(Q)$; then both P and Q have the same number of 1's and -1's, so they have the same length ℓ. Let $j \le \ell$ be the last position at which they differ; interchanging P and Q if necessary, assume $P_j = -1$, $Q_j = 1$. Since P and Q contain the same number of 1's and agree after position j, the excess 1 at position j is cancelled earlier: there is $i < j$ with $P_i = 1$, $Q_i = -1$. As P is branching, its 1-child occurs in the trie, so we can choose $A \in \mathcal{F}$ that starts with P followed by 1. As Q is branching, we can choose $B \in \mathcal{F}$ that starts with Q followed by -1. At coordinates $i, j, \ell+1$:

	i	j	$\ell+1$
A	1	-1	1
B	-1	1	-1

Since P and Q have equally many 1's and -1's, $\sum_{h \le \ell}(A_h - B_h)) = 0$, while coordinate $\ell+1$ contributes 2; as both A and B contain r 1's and $n-r$ -1's, the sum over all n coordinates is also 0. Hence

$$\sum_{h=\ell+2}^{n} (A_h - B_h) = -2. \tag{2}$$

If no coordinate $h > \ell+1$ had $A_h = -1$, $B_h = 1$, every summand in (2) would be 0 or 1, which cannot sum to -2. So such an h exists, and the four indices $i, j, \ell+1, h$ form an illegal configuration, a contradiction. Thus λ is injective, $b \le r(n-r)$, and (1) completes the proof. □

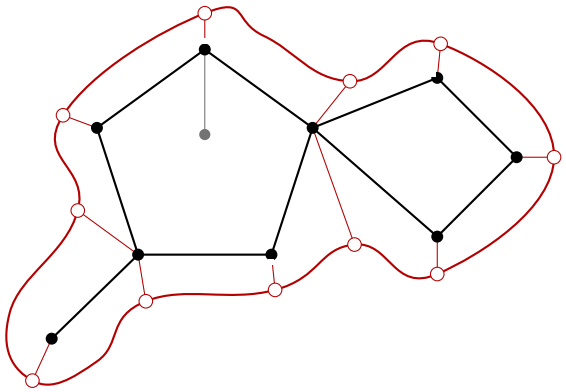

Figure 2: Turning the non simple infinite face f (in black) into a simple face f' (in red).

Proof of Theorem 1. We assume G has no self loops since those do not affect distances. We also assume the designated face f is simple. This can be done without loss of generality by the following transformation. Let W be the boundary cyclic walk of f. Let $s_0, s_2, \ldots, s_{k-1}$ be the vertices of f along W. Note that if W is not a simple cycle then the s_i's are not distinct. For each i, add a new distinct pendant vertex v_i embedded inside f with and edge s_iv_i embedded between $s_{i-1}s_i$ and s_is_{i+1} in the cyclic order around s_i (indices are modulo k). Now connect all the new pendant vertices with edges v_iv_{i+1}. See Fig. 2. This creates a new face f' whose boundary walk is a simple cycle of length k through the newly added vertices. This construction does not change the distance between any two vertices of the original G. Furthermore, For any vertex u of the original G and any $0 \leq i \leq \ell$, the u-to-s_i distance in G equals the u-to-v_i distance in the modified graph. Hence, the number of patterns for f is bounded by the number of patterns for the new simple face f' in the modified graph.

We therefore continue under the assumption that f is simple. We subdivide each edge of G by introducing an artificial vertex in the middle of each edge. This makes the graph bipartite. In particular, the size of the distinguished face f becomes $2k$, and the distance between each pair of original vertices of G doubles. Since each pattern in the subdivided G corresponds to exactly one pattern in the original G, it therefore suffices to bound the number of patterns in the subdivided G. Hence, for the remainder of the proof when we refer to G we mean the subdivided G. Since G is unweighted undirected and bipartite, distances from any vertex u to adjacent vertices on the distiguished face f differ by exactly 1. Hence, each pattern is a vector in $\{-1, +1\}^{2k-1}$.

Denote $s_k = s_0$. Then by cyclicity, for any vertex v, the sum $\sum_{i=1}^{k} d_G(v, s_i) - d_G(v, s_{i-1})$ is 0. Hence, by definition of the pattern p^v, summing all the entries of p^v yields ± 1, so the number of 1 entries in p^v is either $k-1$ or k.

Li and Parter [11, Section 3] proved that the set $\mathcal{P}$ of patterns avoids the illegal configuration. Hence, $\mathcal{P}$ can be partitioned into two disjoint sets of patterns $\mathcal{P}_k \cup \mathcal{P}_{k-1}$, where $\mathcal{P}_k$ (resp., $\mathcal{P}_{k-1}$) contains patterns p with exactly k 1's (resp., $k-1$ 1's) and satisfies Lemma 1 with $n = 2k-1$ and $r = k$ (resp., $r = k-1$). Hence, invoking Lemma 1, we get $p_{\#} = |\mathcal{P}| = |\mathcal{P}_k| + |\mathcal{P}_{k-1}| \leq (k(k-1)+1) + ((k-1)k+1) = 2k^2 - 2k + 2 = O(k^2)$. □

3 The Implication to Diameter Computation

The diameter of a directed weighted planar graph can be computed in truly subquadratic time: Cabello [3] gave the first such algorithm, and the current fastest is the $\tilde{O}(n^{5/3})$ time algorithm of Gawrychowski, Kaplan, Mozes, Sharir and Weimann [7]. For *unweighted undirected* planar graphs no uniform improvement has been known. Abboud, Mozes and Weimann [1] gave algorithms with running times $\tilde{O}(nD^2)$ and $n^{3+o(1)}/D^2$, which beat $\tilde{O}(n^{5/3})$ only when the diameter D is below $n^{1/3}$ or above $n^{2/3}$. In this section we show that the new quadratic pattern bound implies such a uniform improvement. Throughout this section G is simple; parallel edges and loops do not affect distances and may be discarded.

Theorem 2 (Unweighted planar diameter)**.** *There is an $\tilde{O}(n^{8/5})$ time randomized algorithm for computing the diameter, the radius, and all vertex eccentricities of an undirected unweighted n-vertex planar graph.*

Proof. Our algorithm incorporates the notion of patterns into the framework of [3, 7]. In that framework, first an *r-division* is performed in linear time [9]. This is a partition of the graph G into $O(n/r)$ *regions* where each region R contains $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices ∂R (vertices shared by other regions). Then, for each region R, the *Voronoi diagram* data structure of [7] is built in $\tilde{O}(r^2)$ time (so $\tilde{O}(n \cdot r)$ time over all regions). A query to this data structure is a weight assignment $w(\cdot)$ to the boundary vertices ∂R. It returns, in $\tilde{O}(\sqrt{r})$ time, a list indicating for each boundary vertex $u \in \partial R$, the furthest vertex from u in R among all the vertices in u's *Voronoi cell*. The Voronoi cell of u contains all vertices $v \in R$ such that u is the boundary vertex that minimizes the quantity $w(u) + d_R(u, v)$.

The Voronoi diagram data structure is used as follows: For every vertex v of G, for every region R of the r-division, a query is performed on the Voronoi diagram of R with the weights $w(\cdot)$ being the distances in G from v to ∂R. This way, the query reveals the furthest vertex from v in G among all vertices of R (and the maximum such value is returned as G's diameter).[5] Computing the weights $w(\cdot)$ from v to all boundary nodes of all regions takes $\tilde{O}(n/\sqrt{r})$ time [5]. The queries performed for each v require $\tilde{O}(\sqrt{r})$ time per region so $\tilde{O}(n/\sqrt{r})$ time over all regions. The total running time is thus $\tilde{O}(n^2/\sqrt{r} + n \cdot r)$ which is $\tilde{O}(n^{5/3})$ by taking $r = n^{2/3}$.

Our idea is to apply the Voronoi queries on R not for every vertex of G but only for every unique pattern in the graph $G \setminus R \cup \partial R$ with respect to the face ∂R.[6] Specifically, among all vertices with the same pattern, we query the Voronoi diagram of R just for the *representative* vertex v that maximizes $d_{G \setminus R \cup \partial R}(v, s_0)$ where s_0 is an arbitrary chosen vertex of ∂R that defines the starting vertex of the pattern. The observation is (cf. [6, Corollary 9]) that if two vertices have the same pattern with respect to distances in $G \setminus R \cup \partial R$, then they have the same pattern w.r.t. distances in all of G as well. Hence, this choice of representative v guarantees that $d_G(v, u) \leq d_G(v', u)$ for every $u \in R$ and every v' that has the same pattern as v. Thus, for the sake of computing the eccentricity of u, considering just the representative v suffices.

There is another issue however. While the vertices of ∂R lie on a face of the the graph $G \setminus R \cup \partial R$, this face does not necessarily consist of just the $O(\sqrt{r})$ vertices of ∂R, but may

[5] A special treatment is required to handle the specific region R that contains v. It is possible that the path from v to its furthest in R does not touch ∂R. To handle this case, we simply run Dijkstra's algorithm from v inside R after initializing the vertices of ∂R with the aforementioned $w(\cdot)$. Dijkstra takes $\tilde{O}(r)$ time for every v and so $\tilde{O}(n \cdot r)$ time overall.

[6] We assume here that ∂R lies on a single face of $G \setminus R \cup \partial R$. In reality, it may lie on a constant number of faces [9]. Handling this case is standard and is described in [7].

contain many (up to $O(r)$) additional vertices. In the weighted setting, this is easily handled by adding artificial edges of weight infinity that connect just the boundary vertices ∂R. In the unweighted setting however, we cannot add these artificial weighted edges (as then the graph is no longer unweighted and so patterns do not apply). Instead, we will only use the (trivial) fact that there is a *facial walk* in R (a walk that visits all vertices of ∂R) that is of length $k = O(r)$. See Lemma 3 in [6] and the preceding discussion there. In other words, we use the $p_{\#} = O(k^2)$ bound on the number of unique patterns where $O(k^2) = O(r^2)$, not $O((\sqrt{r})^2)$.

Formally, we find the unique patterns in the graph G' obtained from G by removing all vertices of R that are not part of the facial walk of R that contains the boundary vertices ∂R. This guarantees that indeed the boundary vertices ∂R lie on a face of size $O(r)$ of $G \setminus R \cup \partial R$. The randomized algorithm of [12] can find all $O(k^2) = O(r^2)$ patterns in this graph in $\tilde{O}(n)$ time (and can easily find for each pattern the appropriate representative v).

For each unique pattern with representative v, we extract from it the v-to-∂R distances in G'. We then use [5][7] to compute, in $\tilde{O}(\sqrt{r})$ time, the v-to-∂R distances in the entire graph G. Over all $O(r^2)$ unique patterns, this takes $O(r^{2.5})$ time. Finally, for each unique pattern with representative v we query in $\tilde{O}(\sqrt{r})$ time the Voronoi diagram data structure (with weights $w(\cdot)$ being the v-to-∂R distances in G) for the furthest vertex from v in R in each Voronoi cell of the Voronoi diagram. The maximum among those gives us the furthest vertex u_v from v in R. The v-to-u_v distance in G can be obtained by adding the distance from v to the corresponding vertex of ∂R to the distance in R from that vertex to u_v (returned by the query to the Voronoi diagram data structure). Over all $O(r^2)$ unique patterns, this also takes $O(r^{2.5})$ time.

To conclude, there are $O(n/r)$ regions and for each region R we spend $\tilde{O}(n + r^{2.5})$ time so overall $\tilde{O}(n^2/r + n \cdot r^{1.5})$. Taking r to be $n^{2/5}$ gives $\tilde{O}(n^{8/5})$ time for computing the diamter. □

References

[1] Amir Abboud, Shay Mozes, and Oren Weimann. What else can Voronoi diagrams do for diameter in planar graphs? In *Proceedings of the 31st Annual European Symposium on Algorithms (ESA)*, volume 274, pages 4:1–4:20, 2023.

[2] Sujoy Bhore, Jie Gao, Hung Le, Csaba D. Tóth, and Lazar Milenković. Metric Sketching and Dynamic Algorithms for Geometric and Topological Graphs (Dagstuhl Seminar 25212). *Dagstuhl Reports*, 15(5):134–157, 2025.

[3] Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, 15(2):21:1–21:38, 2019. Best paper at SODA 2017.

[4] Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-source shortest paths in embedded graphs. *SIAM J. Comput.*, 42(4):1542–1571, 2013.

[5] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006.

[7] To use [5] we need the distances in R between every pair of vertices of ∂R. These distaces can be computed in $O(r \log r)$ time using the MSSP algorithm [4, 8]. Over all regions, this adds only $O(\frac{n}{r} \cdot r \log r) = \tilde{O}(n)$ to the running time.

[6] Viktor Fredslund-Hansen, Shay Mozes, and Christian Wulff-Nilsen. Truly subquadratic exact distance oracles with constant query time for planar graphs. In *Proceedings of the 32nd International Symposium on Algorithms and Computation (ISAAC)*, volume 212 of *LIPIcs*, pages 25:1–25:12, 2021.

[7] Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. *SIAM J. Comput.*, 50(2):509–554, 2021.

[8] Philip N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 146–155, 2005.

[9] Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proceedings of the 45th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 505–514, 2013.

[10] Bernard Leclerc and Andrei Zelevinsky. Quasicommuting families of quantum Plücker coordinates. In *Kirillov's Seminar on Representation Theory*, volume 181 of *American Mathematical Society Translations, Series 2*, pages 85–108. American Mathematical Society, 1998.

[11] Jason Li and Merav Parter. Planar diameter via metric compression. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 152–163, 2019.

[12] Shay Mozes, Nathan Wallheimer, and Oren Weimann. Improved compression of the Okamura–Seymour metric. In *Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC)*, volume 248, pages 27:1–27:19, 2022.

[13] Haruko Okamura and Paul D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981.

[14] Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13:145–147, 1972.