Engineering Greedy Heuristics and Simulated Annealing Methods for the Median Triangulation Under the Parallel Flip Distance

Jacobus Conradi
BARC, University of Copenhagen, Denmark

Benedikt Kolbe
Hausdorff Center for Mathematics, University of Bonn, Germany

Philip Mayer[1]
Institute of Computer Science, University of Bonn, Germany

Jonas Sauer
Karlsruhe Institute of Technology, Germany

Jack Spalding-Jamieson
Independent, Vancouver, Canada

Abstract

We present our approach for the CG:SHOP 2026 challenge. In this international challenge, the goal was to find a median triangulation for a set of triangulations in the parallel flip reconfiguration graph of all triangulations of an underlying point set. Our simulated-annealing-based approach makes use of two ingredients: a heuristic edge selection for approximating the parallel flip distance of two given triangulations, and a heuristic procedure to generate good initial triangulations.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases triangulation, flip distance, parallel flip distance, heuristic, competition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.106

Category CG Challenge

Supplementary Material *Software (Source Code)*: `https://github.com/PhilipMayer94/jflip` archived at `swh:1:dir:5f40dddfbe834cfc16e09dcdac08d2c61d979042`

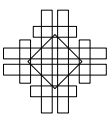
Funding *Jacobus Conradi*: Supported by the Carlsberg Foundation, grant CF24-1929.
Benedikt Kolbe: This work was partially supported by the Lamarr Institute for Machine Learning and Artificial Intelligence.
Philip Mayer: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant FOR-5361 – 459420781.
Jonas Sauer: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant FOR-5361 – 459420781.

Acknowledgements We thank the challenge organizers and other competitors for their time, feedback, and for making this competition and this research possible. We gratefully acknowledge access to the Marvin cluster of the University of Bonn.

1 Introduction

In this paper, we study changing triangulations of a planar point set using *parallel flips*. Given a set of planar triangulations of some point set, we search for a central triangulation and a sequence of parallel flip operations from each input triangulation to the central triangulation, so as to minimize the sum of the number of parallel flip operations in each sequence.

[1] corresponding author



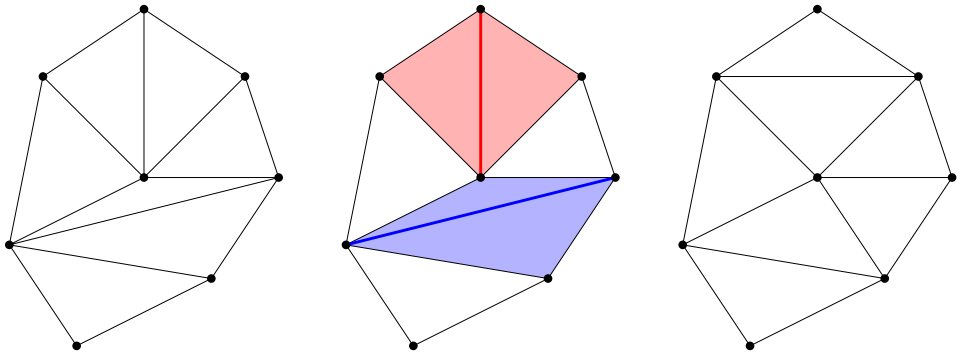


Figure 1 Two edges with their corresponding quadrilateral. They do not share a triangle and can be flipped in parallel, transforming the left triangulation into the right triangulation.

This problem represents the core of the 2026 Computational Geometry Challenge [1] (CG:SHOP), which ranked teams based on their solutions to a provided set of instances. Our team, `jflip2`, was invited to publish its methods after finishing in 4th place with an approach based on heuristic methods inspired by observations for the classical flip distance [5].

For a point set $\mathcal{S} \subset \mathbb{R}^2$, a triangulation $\mathcal{D}$ is a maximal set of non-crossing straight-line edges incident to points in $\mathcal{S}$. In a triangulation, every interior edge e not on the boundary of $\text{conv}(\mathcal{S})$ is adjacent to exactly two triangles. If two such triangles form a convex quadrilateral Q, then we call e *flippable*, and replacing e with the other diagonal $\text{flip}(e)$ results in a valid triangulation $\mathcal{D}' = \mathcal{D} \setminus \{e\} \cup \{\text{flip}(e)\}$. Replacing e with $\text{flip}(e)$ is called a *unit flip*.

A set of edges E in a triangulation $\mathcal{D}$ is called a *parallel flip operation* if each edge $e \in E$ is flippable and no two edges in E are part of the same triangle [4]. Since no two edges in a parallel flip share a triangle, we again obtain that $\mathcal{D}' = \mathcal{D} \setminus E \cup \{\text{flip}(e) \mid e \in E\}$ is a valid triangulation. Parallel flips induce a graph PG, whose vertices are the set $\mathfrak{D}$ of all triangulations, in which two triangulations are adjacent if one can be transformed into the other by exactly one parallel flip. Note that the edges of this graph are undirected, since parallel flips are reversible, i.e., $\text{flip}(\text{flip}(e)) = e$, and, classically, this graph is connected [7]. A sequence of parallel flip operations transforming $\mathcal{D}$ to $\mathcal{D}'$ is a path in PG from $\mathcal{D}$ to $\mathcal{D}'$. The length $\mathrm{d}_{\text{pf}}(\mathcal{D}, \mathcal{D}')$ of a shortest path from $\mathcal{D}$ to $\mathcal{D}'$ in PG is called the *parallel flip distance*.

Given these definitions, we formulate the challenge posed in the CG:SHOP contest.

▶ **Problem 1** (Median Point for Parallel Flip Distance). *Given a set $\mathcal{S}$ of points and a set $\mathfrak{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ of triangulations (terminals), find a triangulation (center) $\mathcal{C}$ and paths $\{P_i\}$ in PG from each $\mathcal{T}_i$ to $\mathcal{C}$, minimizing*

$$\text{cost}(\mathcal{C}, \mathfrak{T}) = \sum_{\mathcal{T}_i \in \mathfrak{T}} |P_i|$$

Ideally, we want $|P_i|$ to be $\mathrm{d}_{\text{pf}}(\mathcal{C}, \mathcal{T}_i)$. This raises the intermediate problem of computing the parallel flip distance of two triangulations. Computing the unit flip distance, where the goal is minimizing the number of unit flips, is NP-hard [8, 10]. The complexity of computing the parallel flip distance remains open. For the unit flip distance, exact methods [9] based on shortest paths and integer linear programs have been employed. To us, these approaches appeared ill-suited for large instances (although the winning team `Shadoks` [3] ultimately employed an optimal SAT solver with great success). Instead, we employ a greedy approach inspired by [5] for the sub-task of approximating $\mathrm{d}_{\text{pf}}(\cdot)$.

Based on this heuristic distance computation, we solve Problem 1 via a simulated-annealing-based approach. Our method is separated into three stages:

1. Compute an initial candidate center $\mathcal{C}$.
2. Improve $\mathcal{C}$ via simulated annealing, with parallel flip distance heuristics as a guide.
3. Use more expensive distance heuristics to improve $\text{cost}(\mathcal{C}, \mathfrak{T})$.

We describe our parallel flip distance heuristics in Section 2 and our simulated-annealing-based center search in Section 3. Finally, we present experimental results in Section 4.

2 Parallel Flip Distance Heuristics

We start with an observation on parallel flips. Given a triangulation $\mathcal{D}$, we define the *conflict graph* $G_{\mathcal{D}}$ as the graph whose vertices correspond to all flippable edges in $\mathcal{D}$, and in which two vertices are adjacent if the corresponding edges are part of the same triangle and hence cannot be part of the same parallel flip. Thus, parallel flips correspond to independent sets in $G_{\mathcal{D}}$. Based on this, the idea of our flip heuristics $H_u(\mathcal{D}_s, \mathcal{D}_t)$ can be summarized as follows:

1. Start with $\mathcal{D}_0 = \mathcal{D}_s$.
2. Compute a utility value $u(e)$ for each flippable edge $e \in \mathcal{D}_i$, such that flipping an edge with high utility is expected to produce a triangulation closer to the given target $\mathcal{D}_t$.
3. Find a maximum-weight independent set (MWIS) heuristically.
4. Apply the corresponding parallel flip operation to obtain a new triangulation $\mathcal{D}_{i+1}$.
5. Repeat Steps 2–4 with $\mathcal{D}_{i+1}$ until the target triangulation is reached.

Typically, the resulting flip distance estimate is asymmetric, i.e., $H_u(\mathcal{D}, \mathcal{D}') \neq H_u(\mathcal{D}', \mathcal{D})$.

Flip Utilities. Hanke et al. [5] observed that the number of proper crossings between edges of two triangulations is an upper bound on the (unit) flip distance. Consequently, they introduced a heuristic that aims to aggressively decrease the number of such crossings. This definition extends nicely to the parallel flip distance, and serves as the basis for our utility function: Let e be an edge connecting two vertices of $\mathcal{S}$, and let $\mathcal{D}_t$ denote the target triangulation. Define $\text{cr}_{\mathcal{D}_t}(e)$ as the number of proper crossings between e and the edges of $\mathcal{D}_t$. For a flippable edge e in $\mathcal{D}$, let $u_H(e, \mathcal{D}, \mathcal{D}_t) = \text{cr}_{\mathcal{D}_t}(e) - \text{cr}_{\mathcal{D}_t}(\text{flip}(e))$, while for non-flippable edges, let $u_H(e, \mathcal{D}, \mathcal{D}_t) = -\infty$. Note that if $\mathcal{D} \neq \mathcal{D}_t$, at least one edge has positive utility [5]. This heuristic is heavily geared towards the unit flip distance: it does not take into account that flipping an edge e blocks all other edges of its two triangles from being added to the current parallel flip. Consequently, we define a natural extension to this heuristic: Let

$$u_{2L}(e, \mathcal{D}, \mathcal{D}_t) = u_H(e, \mathcal{D}, \mathcal{D}_t) - \sum_{z \in \delta(e)} u_H(z, \mathcal{D}, \mathcal{D}_t),$$

where $\delta(e)$ denotes the (flippable) neighbors of e in the conflict graph $G_{\mathcal{D}}$. Our edge selection for the construction of a parallel flip is sequential, so during the selection process, some neighbors of an edge e may already be blocked by previously selected edges. Thus, we can dynamically exclude them from the penalty sum in $u_{2L}(e, \mathcal{D}, \mathcal{D}_t)$. The values cr, u_H, and u_{2L} naturally extend to multiple target triangulations by summing over all terminals.

MWIS Heuristics. Our main MWIS heuristic scans all flippable edges in decreasing order of utility $u(e, \mathcal{D}, \mathcal{D}_t)$ and adds an edge e if the solution remains an independent set and $u(e, \mathcal{D}, \mathcal{D}_t) > 0$. For more accuracy, at the expense of runtime, we refine the initial greedy solution via a local search that swaps selected edges with some of their non-selected neighbors, as long as it increases the total utility. This process ends when no beneficial swaps remain.

Randomization. We introduce randomization into the greedy edge selection algorithm and rerun the randomized parallel flip sequence computation multiple times to increase the overall accuracy. Concretely, given a parameter k (usually $k \leq 50$), we maintain a buffer of the k unblocked edges with the highest utility. At each selection step, one edge from the buffer is selected uniformly at random and added to the parallel flip before updating the buffer.

Engineering. The main runtime cost of the heuristic $H_u(\mathcal{D}_s, \mathcal{D}_t)$ is the initial computation of $\mathrm{cr}_{\mathcal{D}_t}(e)$ for edges $e \in \mathcal{D}_s$ with respect to the target $\mathcal{D}_t$. Here, a brute-force approach combined with an R-tree outperformed a Bentley-Ottmann sweep [2], which requires exact arithmetic for rational crossing points. For small instances ($|\mathcal{S}| \leq 1000$), we speed up computations of the form $H_u(\mathcal{C}, \mathcal{T})$, with current center $\mathcal{C}$ and terminal $\mathcal{T}$, by precomputing a table that contains the number of crossings $\mathrm{cr}_{\mathcal{T}}(e)$ for every possible edge $e \in \binom{\mathcal{S}}{2}$. As the size of this table is quadratic in $|\mathcal{S}|$, we do not compute it for larger instances.

3 Center Search via Simulated Annealing

To optimize the chosen center, we employ simulated annealing [6] with a fixed time horizon t_{total}. Simulated annealing is a local search technique that iteratively explores the solution space by moving to randomly generated neighboring solutions. Improving moves are always accepted, while worsening moves are accepted with a probability $P_{\text{accept}}(\Delta, t)$ that depends on the objective change Δ and the normalized progress $t = t_{\text{current}}/t_{\text{total}}$. This probability is controlled by a temperature parameter that evolves according to a cooling schedule.

Cooling Schedule. We use an exponential cooling schedule that quickly reduces the temperature such that, in the final quarter of the runtime, the algorithm effectively behaves as a greedy local search. Concretely, we set $T_{\text{quarter}} = -\frac{1}{\ln 0.02}$, and $T(t) = T_0 \cdot \left(\frac{T_{\text{quarter}}}{T_0}\right)^{t/0.75}$, ensuring $T(0.75) = T_{\text{quarter}}$. To dampen the impact of large cost increases, we use a sublinear transformation of Δ and accept moves with probability $P_{\text{accept}}(\Delta, t) = \exp\left(-\frac{\Delta^{2/3}}{T(t)}\right)$. With $T_0 = 30$, this yields strong exploration early on and greedy behavior in the final quarter.

Initial Centers. We use four different strategies to generate the initial center:

1. Delaunay: A simple choice for the initial center is the Delaunay triangulation.
2. Path: For each terminal pair $\mathcal{T}, \mathcal{T}' \in \mathfrak{T}$, compute a short heuristic flip path, evaluate all triangulations along the paths and select the best one as the initial center.
3. Intersection: For each edge $e \in \binom{\mathcal{S}}{2}$, count the total number $\mathrm{cr}_{\mathfrak{T}}(e)$ of crossings with all terminals. Construct the center greedily by adding edges in increasing order of $\mathrm{cr}_{\mathfrak{T}}(e)$.
4. Frequency: Each edge e in $\binom{\mathcal{S}}{2}$ is assigned a frequency based on how many terminals contain e. The initial triangulation then results from greedily selecting edges in decreasing order of frequency, using edges with frequency 0 if necessary.

Neighbor Generation. For small instances, where evaluating the flip heuristic is fast, we generate neighbors randomly: We pick a random number k, then we pick k flippable edges uniformly at random and apply the flips in random order, skipping blocked edges.

For larger instances, increased running time of the distance heuristic results in fewer candidates $\mathcal{C}$ being evaluated, so in later stages, we use an informed neighbor generation: We select a random subset $\mathfrak{T}' \subset \mathfrak{T}$, compute $u_H(e, \mathcal{C}, \mathfrak{T}')$ for all flippable edges and select a random greedy flip with a (small) k-buffer. The transition from uninformed to informed

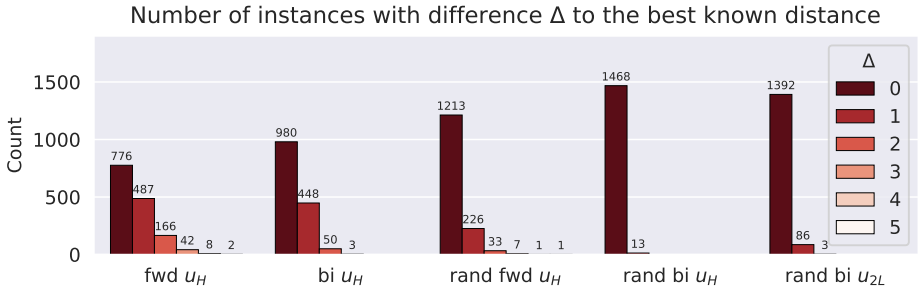


Figure 2 Comparison of deterministic and randomized heuristics: only-forward u_H, bidirectional u_H, only-forward randomized u_H, bidirectional randomized u_H, and bidirectional randomized u_{2L}.

neighbor generation is again controlled by a schedule that gradually increases the probability of selecting an informed neighbor. In the mostly greedy last quarter of the search, we substantially increase the buffer size k for informed flips, reintroducing randomness.

Heuristic Selection. We evaluated the simulated annealing algorithm with many heuristic combinations. The one we found to be the most promising was the following: First, during the exploration phase, we use u_H or u_{2L} to compute only the asymmetric deterministic variants of the heuristics, focusing on computational speed. Then, in the final (greedy) phase of the algorithm, we transition to the full ensemble of heuristics, including all introduced randomization schemes, and their opposites $H_u^{\text{opp}}(\mathcal{D}, \mathcal{D}')$., i.e., $H_u^{\text{opp}}(\mathcal{D}, \mathcal{D}') = H_u(\mathcal{D}', \mathcal{D})$.

4 Experiments

The competition data set consists of 250 instances, grouped into 100 `random`, 101 `woc`, and 49 `rirs` instances. The `random` and `woc` instances are small (15–320 points and 2–20 terminals), whereas the `rirs` instances are substantially larger (500–12 500 points and 20–200 terminals). All experiments were conducted on nodes of the HPC cluster Marvin (University of Bonn), each equipped with 1024 GB RAM and two Intel Xeon Sapphire Rapids 2.10 GHz CPUs.

Parallel Flip Distance Heuristics. To evaluate the quality of the heuristics, we generated up to ten distinct terminal pairs for each instance (1481 samples in total) and computed the parallel flip distance using u_H and u_{2L}, combining greedy and local search to compute the MWIS. The computed flip distances range from 1 to 37 and increase with the instance size.

We observed that the distance values of the heuristics using different utility functions differ in only 6% of instances. Hence, in Figure 2, we focus on u_H. As expected, applying the heuristic in a single direction (H_{u_H}) performs worst, with some outliers deviating by up to five flips from the best known values. Taking the minimum of both directions ($\min(H_{u_H}, H_{u_H}^{\text{opp}})$) increases the number of best values found by 25% and eliminates most of the extreme outliers.

Notably, even the unidirectional randomized variant outperforms the bidirectional deterministic one, yielding a further performance gain of 25%, although some outliers with large deviations from the best value remain. Minimizing over both directions yields the best solutions for most instances; using u_{2L} instead of u_H performs better in only thirteen cases.

Center Search with Simulated Annealing. We select one representative instance from each data set. Figure 3 shows the progression of the objective value during the center search for different configurations. All configurations use u_H for exploration and the full heuristic

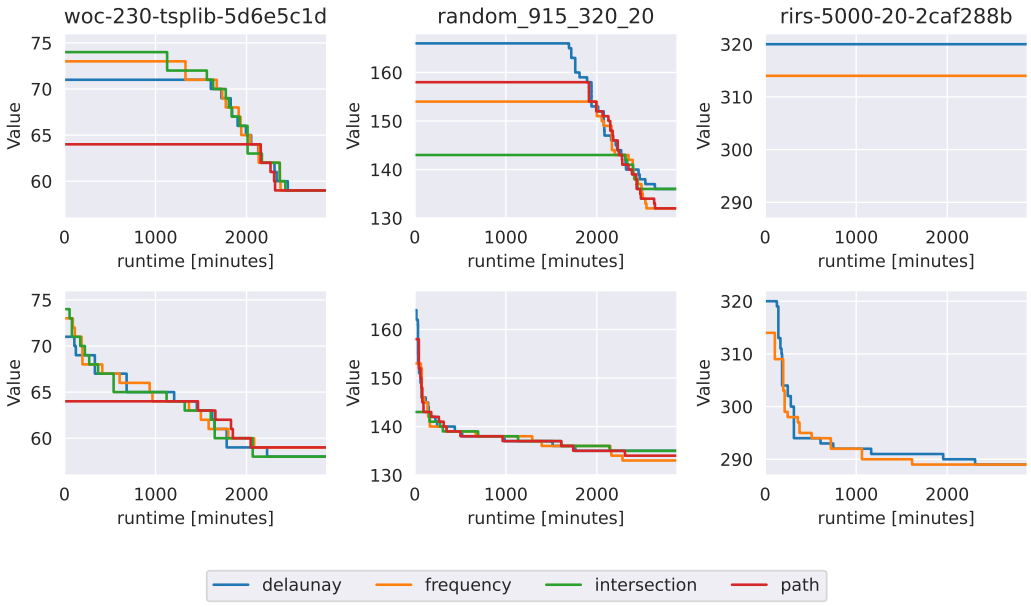


Figure 3 Objective value progression for different configurations during simulated annealing with varying initial centers. Top: uninformed neighbors; bottom: uninformed and informed neighbors.

ensemble in the final quarter. The main difference is in the neighbor selection: the upper row uses only random neighbors, while the bottom row incorporates informed neighbors. As expected, purely random neighbors do not improve the current best value during the exploration phase, but then improve the solution during the final (greedy) search. In contrast, the informed search shows gradual improvements, albeit with a higher risk of getting trapped in local optima. For small instances, the random configuration achieves slightly better final results, but unlike the informed search, it fails to find improvements on the `rirs` instance.

Figure 4 shows that the final objective value changes little with longer runtime for either neighbor selection and can even get worse due to randomization. Further experiments indicate that two to three days is a sweet spot, and that reruns should be favored over extended optimization time. Overall, our example instances are representative: on `random`, uninformed usually wins; on `rirs`, informed always wins; and on `woc`, both outcomes were observed.

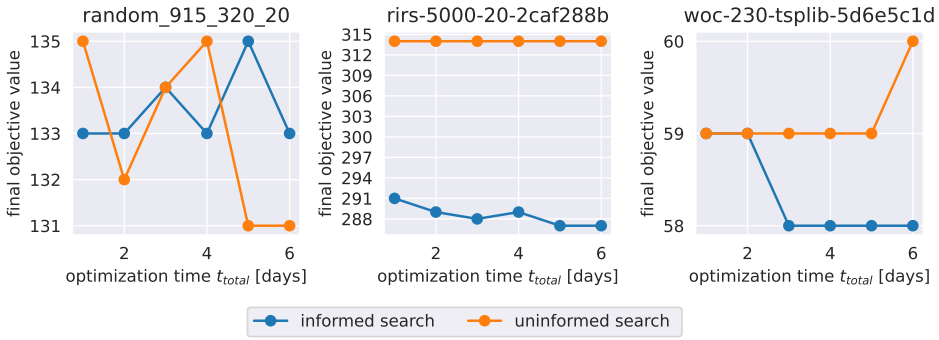


Figure 4 Final objective value after running for t_{total} days starting at the frequency center.

References

1 Oswin Aichholzer, Joseph Dorfer, Sándor P. Fekete, Phillip Keldenich, Peter Kramer, and Stefan Schirra. Central triangulation under parallel flip operations: The cg:shop challenge 2026, 2026. arXiv:2603.18812.

2 Jon L. Bentley and Thomas A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979. doi:10.1109/TC.1979.1675432.

3 Guilherme Dias da Fonseca and Yan Gerard. Shadoks approach to parallel reconfiguration of triangulations. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.107.

4 Jérôme Galtier, Ferran Hurtado, Marc Noy, Stephane Perennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *Int. J. Comput. Geom. Appl.*, 13(2):113–133, 2003. doi:10.1142/S0218195903001098.

5 Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2:570–579, April 1996. doi:10.3217/JUCS-002-08-0570.

6 S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.

7 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972. doi:10.1016/0012-365X(72)90093-3.

8 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. 24th Canadian Conference on Computational Geometry (CCCG'12). doi:10.1016/j.comgeo.2014.11.001.

9 Philip Mayer and Petra Mutzel. Engineering A* Search for the Flip Distance of Plane Triangulations. In *22nd International Symposium on Experimental Algorithms (SEA 2024)*, volume 301 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SEA.2024.23.

10 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.