# Shadoks Approach to Parallel Reconfiguration of Triangulations

**Guilherme D. da Fonseca**
LIS, Aix-Marseille Université, France

**Fabien Feschet**
LIMOS, Université Clermont Auvergne, France

**Yan Gerard**
LIMOS, Université Clermont Auvergne, France

## Abstract

We describe the methods used by Team Shadoks to win the CG:SHOP 2026 Challenge on parallel reconfiguration of planar triangulations. Our approach combines exact methods based on SAT with several greedy heuristics, and also makes use of SAT and MaxSAT for solution improvement.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Exact algorithm, SAT, MaxSAT, heuristic, computational geometry

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2026.107

**Category** CG Challenge

**Related Version** *Full Version*: `https://arxiv.org/abs/2603.21293`

**Supplementary Material** *Software (Source Code)*: `https://github.com/gfonsecabr/shadoks-CGSHOP2026`, archived at `swh:1:dir:14b1868ef49dcd479ff783d738ee448d33be9221`

**Acknowledgements** We would like to thank the Challenge organizers and other competitors for their time, feedback, and making this whole event possible. We would like to thank Hélène Toussaint, Raphaël Amato, Boris Lonjon, and William Guyot-Lénat from LIMOS, as well as the Qarma and TALEP teams and Manuel Bertrand from LIS, who continue to make the computational resources of the LIMOS and LIS clusters available to our research. We would also like to thank Aldo Gonzalez-Lorenzo for the very useful discussion on SAT models.

## 1 Introduction

The CG:SHOP Challenge is an annual competition in geometric optimization. In 2026, the challenge focuses on a reconfiguration problem between planar triangulations. Our team, called *Shadoks*, won first place with the best solution (among the 28 participating teams) to 249 instances out of 250 instances and provably optimal solutions to 189 instances.

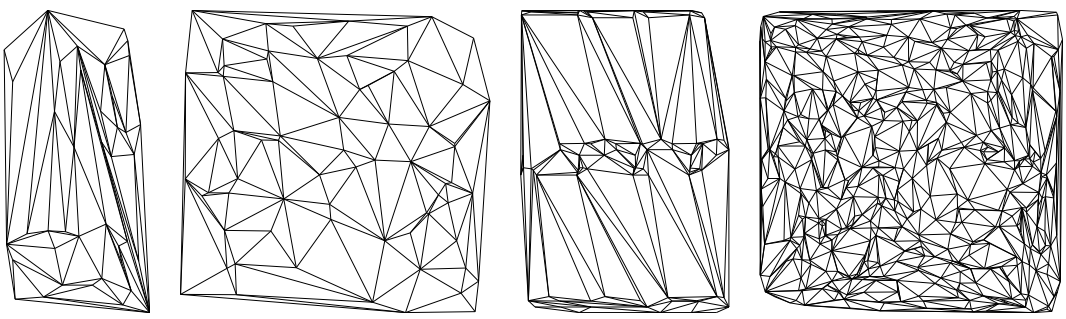

**Figure 1** Our best centers to instances `random_78_40_10`, `woc-70-random-9a7d18d3`, `woc-90-tsplib`, and `rirs-500-50-23d00ec5`, respectively.

In this paper, we outline the exact methods that we employed. The heuristic solvers are only presented in the full version due to space limitations. We start with some definitions that allow us to describe the problem. Throughout, we consider triangulations of a common point set $S \subset \mathbb{R}^2$.

Given a triangulation $T$, a *unit flip* is the operation that removes an edge $e \in T$ and adds an edge $e'$. The unit flip of the edge $e = uv$ considers the empty convex quadrilateral $u, u_2, v, v_2$ and replaces its diagonal $uv$ it by the other diagonal $u_2v_2$ (Figure 2).

Similarly, a *parallel flip* removes a set of edges $E \subset T$ and adds a set of edges $E'$, in a way that $T' = T \setminus E \cup E'$ is a triangulation, with the condition that no two edges of $E$ are in the same triangle in $T$. A *path* of *length* $\ell$ is a sequence of triangulations $T_0, \ldots, T_\ell$ such that for all $i$, the triangulation $T_{i+1}$ is obtained from $T_i$ by performing a parallel flip.

An *instance* is a set $S \subset \mathbb{R}^2$ of $n$ points and a set $\mathcal{T} = \mathbf{T}_1, \ldots, \mathbf{T}_{|\mathcal{T}|}$ of triangulations of $S$, called *input triangulations*. A *solution* is a set of paths $P_1, \ldots, P_{|\mathcal{T}|}$ such that $P_i$ starts at $\mathbf{T}_i$ for all $i$ and all paths end in a common triangulation called *center*. The goal is to find a solution that minimizes the *objective value* defined as the sum of the lengths of its paths.

During the competition, the organizers provided a total of 250 instances, with $n$ ranging from 15 to 12,500 points and $|\mathcal{T}|$ ranging from 2 to 200 triangulations. The 250 instances are divided into three classes: 100 `random` instances, 101 `woc` instances, and 49 `rirs` instances. The former two instances have up to 320 points and 2 to 20 input triangulations (hence, we call them `small` instances), while the latter have 500 to 12500 points and 20 to 200 input triangulations. The centers of some of our best solutions are presented in Figure 1. Additional details about the challenge can be found in the organizers' survey paper [2].

Our best solvers heavily rely on the SAT solver `CaDiCal` [5] and the MaxSAT solver `EvalMaxSAT` [3]. Nevertheless, we also developed heuristics that do not rely on any external solver, which are important to find initial solutions to large instances, which are then improved by roughly 10% using SAT and MaxSAT solvers., which are presented in the full version.

Other team strategies include SAT solvers for small instances [4], heuristics [9], and local search and simulated annealing [7] to improve solutions.

## 2 Exact Algorithms

### 2.1 Path SAT Formulation

Next, we describe a SAT formulation for the following decision problem. The input is a set $S$ of $n$ points, an integer $\ell$ and two triangulations $T_0, T_\ell$. The output is whether there exists a path $T_0, \ldots, T_\ell$ of length $\ell$.

We define two types of variables. For $i = 0, \ldots, \ell$ and for $u \neq v \in S$, we define an *edge variable* $e(u, v, i)$. The variable $e(u, v, i)$ represents that the edge $uv$ is in the triangulation $T_i$. There are $O(n^2\ell)$ edge variables. It would be possible to define a SAT formulation using only such variables. However, a SAT formulation that performed much better in our experiments uses a second type of variable.

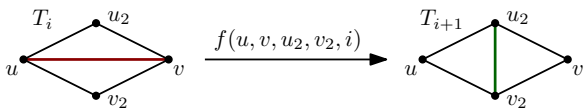


**Figure 2** Illustration of a flip and the associated variable $f(u, v, u_2, v_2, i)$.

We say that a convex quadrilateral is *empty* if it contains no point of $S$ except for its vertices. For $i = 0, \ldots, \ell$-1 and for an empty convex quadrilateral $u, u_2, v, v_2$, we introduce *flip variables* $f(u, v, u_2, v_2, i)$, which is true if and only if $uv$ is in triangulation $T_i$ and $u_2v_2$ is in triangulation $T_{i+1}$, as shown in Figure 2. Notice that if the points are uniformly distributed, then the number of empty convex quadrilaterals is $\Theta(n^2)$ [8], which means that for uniformly distributed points, the number of flip variables is also $O(n^2\ell)$. However, the number of flip variables is $\Theta(n^4\ell)$ if the points are in convex position (which is not the case for the challenge instances). Next, we describe the different types of clauses.

**Start and target.** For every edge variable $e(u, v, 0)$, we have the clause $e(u, v, 0)$ if $uv \in T_0$ and $\neg e(u, v, 0)$ if $uv \notin T_0$. For every edge variable $e(u, v, \ell)$, we have the clause $e(u, v, \ell)$ if $uv \in T_\ell$ and $\neg e(u, v, \ell)$ if $uv \notin T_\ell$.

**Flips need edges.** For every flip variable $f(u, v, u_2, v_2, i)$, we have the 5 binary CNF clauses translating the implication

$$f(u, v, u_2, v_2, i) \implies e(u, v, i) \wedge e(u, v_2, i) \wedge e(u, u_2, i) \wedge e(v, v_2, i) \wedge e(v, u_2, i).$$

**Flips keep edges.** Similarly, or every flip variable $f(u, v, u_2, v_2, i)$, we have the 5 binary CNF clauses translating the implication

$$f(u, v, u_2, v_2, i) \implies e(u_2, v_2, i{+}1) \wedge e(u, v_2, i{+}1) \wedge e(u, u_2, i{+}1) \wedge e(v, v_2, i{+}1) \wedge e(v, u_2, i{+}1).$$

**Flips flip edges.** For every flip variable $f(u, v, u_2, v_2, i)$, we have the binary clauses translating

$$f(u, v, u_2, v_2, i) \implies \neg e(u, v, i + 1).$$

**Edge changes require flips.** The last type of clause is the only one that has more than 2 variables in CNF form. It states that if the edge variable changes from triangulation $i$ to $i + 1$, then there must be a flip. The $\bigvee$ below considers all values of the subscript that define existing flip variables. We have two such clauses for each edge variable:

$$e(u, v, i) \wedge \neg e(u, v, i + 1) \implies \bigvee_{u_2, v_2} f(u, v, u_2, v_2, i) \text{ and}$$

$$\neg e(u_2, v_2, i) \wedge e(u_2, v_2, i + 1) \implies \bigvee_{u, v} f(u, v, u_2, v_2, i).$$

**Eliminating variables and clauses.** The number of variables and clauses grows very fast, even though the number of clauses is linear in the number of variables. Next, we show how to eliminate many variables from the model. All eliminated variables are defined as *false* and the clauses that become tautologies are eliminated. If a CNF clause becomes empty, then the problem is unsatisfiable.

The following theorem is easy to prove and implies that $\Omega(\log n)$ parallel flips are sometimes necessary to reconfigure two triangulations of $n$ points, even when the points are in convex position. We say that two segments *cross* if they intersect at a point that is not an endpoint of either segment.

▶ **Theorem 1.** *Consider two triangulations $T, T'$ of $S$ such that a parallel flip transforms $T$ into $T'$ and a segment $s$ with endpoints in $S$. Let $\chi, \chi'$ respectively denote the number of edges of $T, T'$ crossed by $s$. We then have $\chi' \geq \lfloor \chi/2 \rfloor$.*

Consequently, we only define the variable $e(u, v, i)$ when $uv$ crosses strictly less than $2^i$ edges of $T_0$ and strictly less than $2^{\ell-i}$ target edges. We only define flip variables when a certain set of edge variables is defined: $f(u, v, u_2, v_2, i)$ is only defined when $uv, uv_2, uu_2, vv_2, vu_2$, are all defined at $i$ and $u_2v_2, uv_2, uu_2, vv_2, vu_2$, are all defined at $i + 1$.

## 2.2 Solution SAT Formulation

Next, we describe a SAT formulation for the following decision problem. Recall that an instance is a set $S$ of $n$ points and a list $\mathcal{T}$ of input triangulations $\mathbf{T}_1, \ldots, \mathbf{T}_{|\mathcal{T}|}$. The input of the decision problem is an instance and $|\mathcal{T}|$ integers $\ell_1, \ldots, \ell_{|\mathcal{T}|}$. The output is whether there exists a solution $P_1, \ldots, P_{|\mathcal{T}|}$ such that path $P_i$ has length $\ell_i$ for all $i$.

We model the $|\mathcal{T}|$ paths $P_1, \ldots, P_{|\mathcal{T}|}$ independently as before, starting path $P_i$ at the input triangulation $\mathbf{T}_i$. The final triangulation of each path is unknown, but the same edge variables are used for the final triangulation of every path, since a valid solution requires that all paths end in the same triangulation. It is easy to see that the SAT formulation is satisfiable if and only if there exists a solution with the given lengths.

## 2.3 Lower Bound

In order to obtain an exact solution to an instance $\mathcal{I}$, we start by computing a lower bound to its objective value. We say that the *distance* between two triangulations $T, T'$ is the length of the shortest path from $T$ to $T'$. We create a complete directed graph $G(\mathcal{I})$ with edge lengths as follows. The vertices are the triangulations $\mathcal{T}$ and the length of each edge is the distance between the corresponding triangulations. A *cycle packing* of $G$ is a collection of vertex-disjoint directed cycles, i.e. a subset of edges such that each vertex has at most one outgoing and at most one incoming edge in the subset. The graph is directed to allow for cycles with only 2 edges. The *length of a cycle* is the sum of the lengths of its edges, and the *length of a cycle packing* is the sum of the lengths of its cycles. The maximum length cycle packing can be solved in polynomial time using a reduction to maximum weight bipartite matching [6]. The following theorem is easy to show.

▶ **Theorem 2.** *Given an instance $\mathcal{I}$, the objective value of a solution is at least the length of any cycle packing of $G(\mathcal{I})$ divided by 2.*

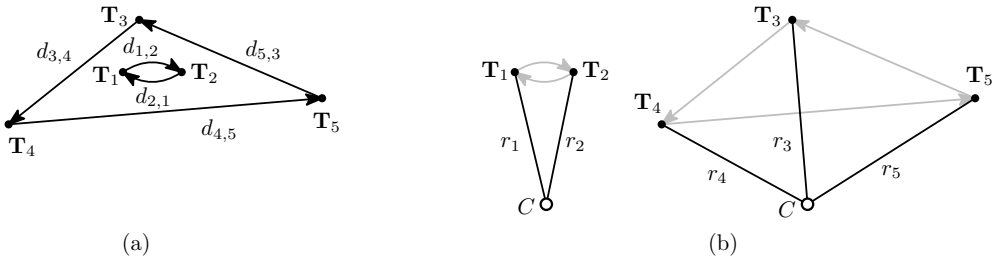


**Figure 3** (a) A cycle packing. (b) Illustration of the proof. In this example,$d_{1,2} \leq r_1 + r_2$, $d_{2,1} \leq r_2 + r_1$, $d_{3,4} \leq r_3 + r_4$, $d_{4,5} \leq r_4 + r_5$, and $d_{5,3} \leq r_5 + r_3$ by triangle inequality.

## 2.4 The Exact Solver

First, we use the exact path formulation from Section 2.1 to calculate the distance between all $\binom{\mathcal{T}}{2}$ pairs of input triangulations using a SAT solver (in our case, `CaDiCal` [5]). It is easy to formulate the problem of finding a maximum length cycle packing as a weighted MaxSAT problem, which provides a lower bound $b$ to the objective value (see Section 2.3). We solve this problem using a weighted MaxSAT solver (in our case `EvalMaxSAT` [3]). We then use backtracking to list all integer solutions to $\ell_0, \ldots, \ell_{|\mathcal{T}|} = b$ that satisfy $\ell_i + \ell_j \geq \text{distance}(T_i, T_j)$. We use the SAT formulation from Section 2.2 to test the existence of a solution with the given lengths $\ell_0, \ldots, \ell_{|\mathcal{T}|}$, again using the `CaDiCal` SAT solver. If a solution is found, then it is optimal. Otherwise, we increment $b$ and repeat. Notice that $b$ is always a lower bound to the objective value. Hence, if a solution obtained by a heuristic attains this lower bound, then it is optimal.

## 2.5 Happy Edges Conjecture

The happy edges conjecture [1] is a general conjecture that is *false* for some reconfiguration problems and *true* for others.

▶ **Conjecture 3.** *For any pair of configurations $T, T'$, there exists a shortest path between $T, T'$ where the edges that are common to both $T$ and $T'$ appear in all intermediate configurations.*

The conjecture is *false* for triangulations under unit flips and arbitrary points [10] but *true* when the points are in convex position [11]. Our experiments lead us to believe that the conjecture is *true* for parallel flips, as we could not find a counterexample.

When computing a path of length $\ell$ from $T_0$ to $T_\ell$ using SAT, for every edge $uv$ that appears in both $T_0$ and $T_\ell$, we add clauses $e(u, v, i)$ that force the variable to be true for all $i$. More importantly, we then eliminate every edge variables corresponding to edges that cross $uv$. The same idea can be applied to the SAT formulation that finds a solution, but then only the edges that appear in all input triangulations are forced to be true for all $i$, and again the edges that cross them are eliminated.

Furthermore, when computing a path, for every edge $uv \in T_\ell$, we eliminate flip variables that remove $uv$, i.e. $f(u, v, u_2, v_2, i)$ for all $u_2, v_2, i$. Similarly, for every edge $u_2v_2 \in T_0$, we eliminate flip variables that insert $u_2v_2$, that is $f(u, v, u_2, v_2, i)$ for all $u, v, i$.

# 3 Results

In this section, we present the computational results that we obtained with our implementation of the aforementioned algorithm. In Section 3.1, we present the results on computing short paths between two given triangulations. In Section 3.2, we present our exact solver, with and without the happy edges conjecture.

The solvers were coded in C++ and compiled with GCC and run a single thread. During the competition, they were executed on several Linux computers, either using GNU Parallel [12] for local executions or Slurm [13] for cluster executions. It was very useful to have access to machines with 128GB or more RAM to solve large SAT formulations with `CaDiCal` [5], which has been able to solve SAT instances with more than 50 million variables and 500 million clauses. The time measurements on this paper use an AMD Ryzen 9 9900X CPU and ASUS TUF B650M motherboard with 128GB of RAM running Fedora Core 43.

**Table 1** Length and computation time (in milliseconds) for a path from the Delaunay triangulation to the first input triangulation of the `rirs-`$n$`--20` instance for different values of the number of points $n$. The Heuristic column shows the smallest length obtained by applying four heuristics (greedy (forward/backward) and squeaky-wheel (forward/backward)) described in the full version of the paper and the total time to run the four heuristics. The SAT columns correspond to SAT formulation with or without the happy edges conjecture, unless it takes too long.

| | Heuristic | | SAT happy | | SAT exact | |
|---|---|---|---|---|---|---|
| $n$ | length | total time | length | time | length | time |
| 500 | **12** | 70.9 | **12** | 1512 | **12** | 10191 |
| 1000 | 12 | 77.6 | **11** | 6806 | **11** | 78264 |
| 1500 | 12 | 100 | **11** | 19302 | **11** | 201353 |
| 2000 | **13** | 208 | **13** | 16937 | **13** | 485177 |
| 3000 | **14** | 695 | **14** | 140729 | · | · |
| 4000 | **15** | 1340 | **15** | 101402 | · | · |
| 5000 | **16** | 656 | **16** | 1037840 | · | · |
| 6000 | **16** | 1891 | **16** | 271081 | · | · |
| 7000 | 17 | 777 | **16** | 471389 | · | · |

## 3.1 Path Calculation

Computing short paths between two given triangulations is a key component to obtain good solutions. Typically, these paths are computed with an input triangulation as one extreme, and a triangulation that makes a reasonably good center as the other extreme. Table 1 shows the best length of the path obtained with our heuristics (described in the full version of the paper) with the associated running times compared to our SAT formulation with and without the happy edges conjecture. The paths are computed from the Delaunay triangulation that makes a reasonably good (but not very good) center to the first input triangulation of several instances. The SAT paths are obtained by first running a heuristic in both directions, and then iteratively decreasing the path length with our SAT solution.

## 3.2 Exact Solutions

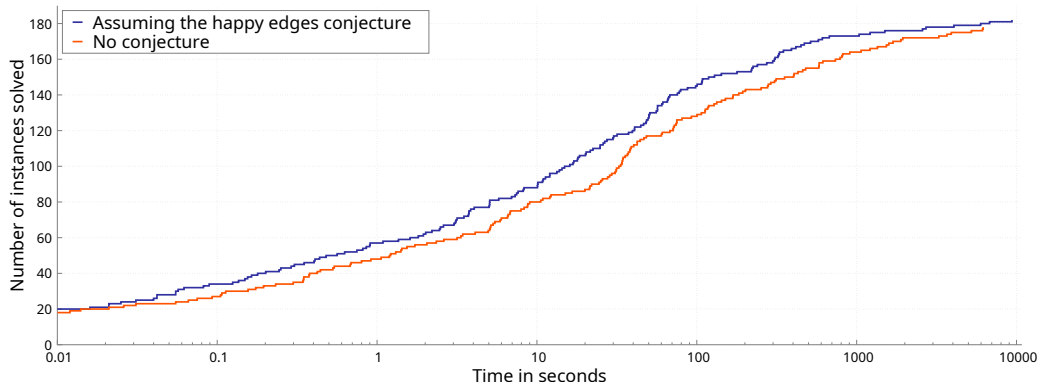


**Figure 4** Number of exact solutions found as a function of the running time over 3 hours of execution with and without assuming the happy edges conjecture.

We end the paper with the performance of our exact solver. Figure 4 plots the number of exact solutions found as a function of runtime, with and without the happy edges conjecture. Since the solution values are identical in both cases, the happy edge conjecture holds.

## References


1. Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees, 2022. doi:10.48550/arXiv.2206.03879.
2. 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.
3. Florent Avellaneda. A short description of the solver EvalMaxSAT. *MaxSAT Evaluation*, 8:364, 2020.
4. Lorenzo Battini and Marko Milenković. ETH flippers approach to parallel reconfiguration of triangulations: SAT formulation and heuristics. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.105.
5. Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In *Computer Aided Verification - 36th International Conference, CAV*, volume 14681 of *LNCS*, pages 133–152, 2024. doi:10.1007/978-3-031-65627-9_7.
6. P Biró, D Manlove, and R Rizzi. Maximum weight cycle packing in optimal kidney exchange programs, university of glasgow, department of computing science. Technical report, Technical Report TR–2009–298, 2009.
7. Jacobus Conradi, Benedikt Kolbe, Philip Mayer, Jonas Sauer, and Jack Spalding-Jamieson. Engineering greedy heuristics and simulated annealing methods for the median triangulation under the parallel flip distance. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.106.
8. Ruy Fabila-Monroy, Clemens Huemer, and Dieter Mitsche. The number of empty four-gons in random point sets. *Electronic Notes in Discrete Mathematics*, 46:161–168, 2014. doi:10.1016/j.endm.2014.08.022.
9. Jaegun Lee, Seokyun Kang, Hyeonseok Lee, Hyeyun Yang, and Taehoon Ahn. CG#Hunters approach to central triangulation under parallel flip operations. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.108.
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.
11. Daniel D. Sleator and William P. Tarjan, Robert E.and Thurston. Rotation distance, triangulations, and hyperbolic geometry. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 122–135, 1986.
12. O. Tange. GNU parallel – The command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, February 2011. URL: http://www.gnu.org/s/parallel.
13. Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60, 2003. doi:10.1007/10968987_3.