Reconfiguration problems

by

Zuzana Masárová

June, 2020

A thesis presented to the
Graduate School
of the
Institute of Science and Technology Austria, Klosterneuburg, Austria
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

IST AUSTRIA
Institute of Science and Technology

The thesis of Zuzana Masárová, titled *Reconfiguration Problems*, is approved by:

Supervisor: Uli Wagner, IST Austria, Klosterneuburg, Austria

Signature: ____________________

Co-supervisor: Herbert Edelsbrunner, IST Austria, Klosterneuburg, Austria

Signature: ____________________

Committee Member: Anna Lubiw, University of Waterloo, Waterloo, Canada

Signature: ____________________

Committee Member: Krzysztof Pietrzak, IST Austria, Klosterneuburg, Austria

Signature: ____________________

Defense Chair: Edouard Hannezo, IST Austria, Klosterneuburg, Austria

Signature: ____________________

signed page is on file

IST Austria Thesis, ISSN: 2663-337X

ISBN: 978-3-99078-005-3

I hereby declare that this thesis is my own work and that it does not contain other people's work without this being so stated; this thesis does not contain my previous work without this being stated, and the bibliography contains all the literature that I used in writing the dissertation.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

I certify that any republication of materials presented in this thesis has been approved by the relevant publishers and co-authors.

Signature: ______________________

Zuzana Masárová

June, 2020

signed page is on file

Abstract

This thesis considers two examples of reconfiguration problems: flipping edges in edge-labelled triangulations of planar point sets and swapping labelled tokens placed on vertices of a graph. In both cases the studied structures – all the triangulations of a given point set or all token placements on a given graph – can be thought of as vertices of the so-called reconfiguration graph, in which two vertices are adjacent if the corresponding structures differ by a single elementary operation – by a flip of a diagonal in a triangulation or by a swap of tokens on adjacent vertices, respectively. We study the reconfiguration of one instance of a structure into another via (shortest) paths in the reconfiguration graph.

For triangulations of point sets in which each edge has a unique label and a flip transfers the label from the removed edge to the new edge, we prove a polynomial-time testable condition, called the Orbit Theorem, that characterizes when two triangulations of the same point set lie in the same connected component of the reconfiguration graph. The condition was first conjectured by Bose, Lubiw, Pathak and Verdonschot. We additionally provide a polynomial time algorithm that computes a reconfiguring flip sequence, if it exists. Our proof of the Orbit Theorem uses topological properties of a certain high-dimensional cell complex that has the usual reconfiguration graph as its 1-skeleton.

In the context of token swapping on a tree graph, we make partial progress on the problem of finding shortest reconfiguration sequences. We disprove the so-called Happy Leaf Conjecture and demonstrate the importance of swapping tokens that are already placed at the correct vertices. We also prove that a generalization of the problem to weighted coloured token swapping is NP-hard on trees but solvable in polynomial time on paths and stars.

Acknowledgments

I am grateful to my supervisors for the support I have received throughout my doctoral studies and especially to Prof Anna Lubiw who, despite being on another continent, informally co-supervised me through weekly video chat meetings during my entire degree. Also, the work on the Orbit Theorem (Chapter 2) was initiated during my visit to University of Waterloo in July 2015 and the work on Token Swapping (Chapter 3) during a visit in January 2018.

List of Publications

The following publications are included in this thesis:

[i] Anna Lubiw, Zuzana Masárová, and Uli Wagner. A proof of the Orbit Conjecture for flipping edge-labelled triangulations. Discrete & Computational Geometry, 61(4):880-898, Jun 2019.

[ii] Ahmad Biniaz, Kshitij Jain, Anna Lubiw, Zuzana Masárová, Tillmann Miltzow, Debajyoti Mondal, Anurag Murty Naredla, Josef Tkadlec, and Alexi Turcotte. Token swapping on trees. CoRR, abs/1903.06981, 2019.

Chapter 2 of this thesis is joint work with Anna Lubiw and Uli Wagner. Most of this work appeared in [i].

The talk entitled ‘A proof of the Orbit Conjecture for flipping edge-labelled triangulations’ at Symposium on Computational Geometry’17 received the Best Student Presentation Award.

Chapter 3 of this thesis is joint work with Ahmad Biniaz, Kshitij Jain, Anna Lubiw, Tillmann Miltzow, Debajyoti Mondal, Anurag Murty Naredla, Josef Tkadlec, and Alexi Turcotte. The results presented in Chapter 3 cover those sections of [ii] in which I, Zuzana Masárová, was significantly involved. The only exception is Section 3.6 which is very related, hence I provide a sketch of the result and point the reader to the full version of the paper.

Table of Contents

List of Tables

List of Figures

1 Introduction to Reconfiguration Problems

Problems of turning one configuration into another using a limited repertoire of steps have a long history, for example in the study of puzzles and permutation groups [40]. A classic problem is the *15-Puzzle* that consists of a 4×4 playing board covered by 15 numbered square tiles and one empty square. The goal is to arrange the numbers by sliding the tiles.

The puzzle can be generalized to an arbitrary graph as the playing board, some number of tokens placed on its vertices, and a rule determining when a token can slide to another vertex along a graph edge. Characterizing which token configurations are reachable from an initial token placement by following the given rule, or in how many steps they can be reached, are intriguing problems that have been studied by communities in mathematics, computer science and engineering. Tokens on graphs can serve as abstract models for various practical problems like robot motion planning or packet switching.

Another example that demonstrates the importance of local changes is triangulation flipping. In order to construct a triangulation with particular properties, such as one would need when modelling surfaces, it often seems easier if the algorithm starts from an arbitrary triangulation and, by a sequence of local changes (called flips) eventually reaches a required triangulation; rather than to construct the special triangulation right away.

There are many other examples of similar flavour, sometimes in seemingly unrelated settings: recolouring graphs one vertex at a time; changing variable assignments, one value at a time in satisfiability problems; modifying one spanning tree / matching / matroid into another one by local changes; string editing and others; to name a few examples.

Recently, the name *reconfiguration* has been applied to these problems. Reconfiguration problems can be formulated in terms of a *reconfiguration grap*h that has a vertex for each configuration and an edge for each possible reconfiguration step. In case of the 15-puzzle, the vertices are all possible configurations of the 15 numbered tokens on the playing board and two configurations are adjacent if one can be obtained from the other by sliding a single token into a neighbouring empty space.

The general questions that are considered in reconfiguration problems are: can any configuration be reconfigured to any other (connectivity); what is the worst case number of steps required (diameter); and what is the complexity of computing the minimum number of steps required to get from one given configuration to another given configuration (distance).

This thesis develops results in two of the reconfiguration settings: flipping triangulations of 2-dimensional point sets; and token swapping on trees.

Triangulation flipping. Fix a set P on n points in the plane. A *triangulation* of P is a maximal set of pairwise non-crossing line segments, also called edges, whose end vertices are the points of P. The edges subdivide the convex hull of P into triangular regions, hence the name. A *flip* deletes an edge e whose removal leaves a convex quadrilateral, and replaces e by the opposite diagonal of the quadrilateral. The reconfiguration graph in this setting is known as the *flip graph*. It is well known that the flip graph is connected, i.e. any triangulation of a point set can be reconfigured to any other triangulation by some sequence of flips.

We explore the connectivity question in the setting where each edge of a triangulation has a label, and a flip transfers the label of the removed edge to the new edge. It is not true that every labelled triangulation of a point set can be reconfigured to every other labelled triangulation via a sequence of flips, but we characterize when this is possible. There is an obvious necessary condition: for each label l, if edge e has label l in the first triangulation and edge f has label l in the second triangulation, then there must be some sequence of flips that moves label l from e to f, ignoring all other labels. Bose, Lubiw, Pathak and Verdonschot formulated the “Orbit Conjecture”, which states that this necessary condition is also sufficient, i.e. that *all* labels can be simultaneously mapped to their destination if and only if *each* label individually can be mapped to its destination. We prove this conjecture. Furthermore, we give a polynomial-time algorithm to find a sequence of flips to reconfigure one labelled triangulation to another, if such a sequence exists, and we prove an upper bound of $O(n^7)$ on the length of the flip sequence.

Our proof uses the topological result that the sets of pairwise non-crossing edges on a planar point set form a simplicial complex that is homeomorphic to a high-dimensional ball (this follows from a result of Orden and Santos; we give a different proof based on a shelling argument). The dual cell complex of this simplicial ball, called the *flip complex*, has the usual flip graph as its 1-skeleton. We use properties of the 2-skeleton of the flip complex to prove the Orbit Conjecture.

We also provide a modified version of the proof to show that the Orbit Theorem applies to triangulations in which a subset of edges is forbidden from flipping and, consequently, also to triangulations of simple polygons.

Finally, the chapter on triangulations closes by considering examples in which *happy edges* – i.e. edges in a triangulation that already have the correct position and label – must be flipped in order to obtain a shortest flipping sequence between a pair of triangulations. This brings us to analogous problems as are considered in the next chapter on token swapping.

Token swapping on trees. The input to the token swapping problem is a graph with vertices $v_1, v_2, \ldots, v_n$, and n tokens with labels $1, 2, \ldots, n$, one on each vertex. The goal is to get token i to vertex v_i for all $i = 1, \ldots, n$ using a minimum number of *swaps*, where a swap exchanges the tokens on the endpoints of an edge.

The set of moves generate a permutation group and so the reconfiguration graph in this setting is the Cayley graph. In this chapter we concentrate on distance questions on the reconfiguration graph, i.e. on computing optimal token swapping sequences to transform one token configuration into another.

The token swapping problem on general graphs was shown to be NP-complete [13]. Token swapping on a tree, also known as "sorting with a transposition tree," is not known to be in P nor NP-complete. We present some partial results:

1. An optimum swap sequence may need to perform a swap on a leaf vertex that

has the correct token (a *happy leaf*), disproving a conjecture of Vaughan. In fact, we give an example where swapping the happy tokens/leaves saves a constant fraction of swaps as compared to any sequence that would fix them.

2. A generalized problem—weighted coloured token swapping—is NP-hard on trees, but solvable in polynomial time on paths and stars. In this version, tokens and vertices have colours, and colours have weights. The goal is to get every token to a vertex of the same colour, and the cost of a swap is the sum of the weights of the two tokens involved.

Before describing our results in Chapters 2 (Triangulation Reconfiguration) and 3 (Token Swapping), the remainder of this chapter will give a survey of the general reconfiguration framework and some examples of reconfiguration and their applications in other fields.

1.1 Reconfiguration framework

The many examples of reconfiguration mentioned in the previous section were first unified into a common framework in a paper by Ito et al. [85] in 2011. Since then the goal has been to formalize general patterns behind the problems, unify a set of techniques to study the problems, as well as to derive general results that apply to classes of problems across the different contexts. This section provides a brief introduction to the framework. It is based on the surveys by van den Heuvel [137] and Nishimura [111].

In full generality, a *reconfiguration problem* specifies a *reconfiguration graph* and a problem related to the graph. The reconfiguration graph captures relations among certain allowed configurations, given a specific transformation rule. More precisely, the vertices of a reconfiguration graph are all *feasible configurations* and two such configurations are adjacent if one can be obtained from the other by a single application of the transformation rule. What counts as a feasible configuration is often determined by a source problem. Consequently, configurations are sometimes called *feasible solutions* [111].

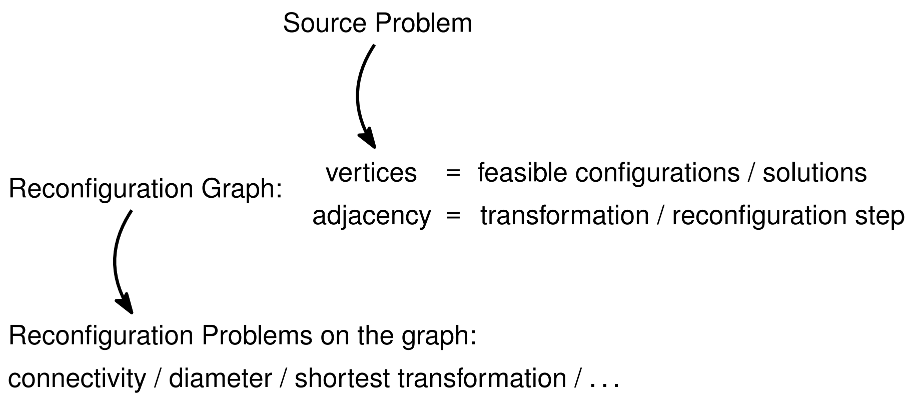

Source Problem
Reconfiguration Graph:
vertices = feasible configurations / solutions
adjacency = transformation / reconfiguration step
Reconfiguration Problems on the graph:
connectivity / diameter / shortest transformation / ...

Figure 1.1: Reconfiguration framework.

For example, in Independent Set Reconfiguration under the token sliding (TS) rule, the source problem is the Independent Set Problem. Recall that given a graph, an *independent set* is a subset of its vertices such that no two vertices are adjacent to each other. Given a graph G and a number k, the *Independent Set Problem* is the decision problem to determine whether or not G contains an independent set of size k. The feasible configurations are then all independent sets of G of size k. These can be represented as token configurations, by placing k tokens on the respective k independent vertices of G. The *token sliding* rule specifies that two configurations are adjacent in the reconfiguration graph if one can be obtained from the other by sliding a single token along an edge to an adjacent empty vertex.

Another example of a reconfiguration graph is the flip graph of triangulations for a planar point set. Here the source problem is to triangulate a point set P (i.e., to partition the convex hull of P into triangles whose vertices are all of points in P), the feasible configurations are all triangulations of P and two triangulations are adjacent if one can be obtained from the other by a single flip, as defined earlier in the introduction.

Note that reconfiguration graphs can be formed based on both tractable and intractable source problems: while the Independent Set Problem is NP-complete, there are polynomial time algorithms to triangulate a point set [50]. Also a trivial source problem can result in a reconfiguration graph with interesting properties. For example, as in Token Swapping, where the feasible configurations are all permutations of tokens $1, \ldots, n$ on n vertices of a graph and the reconfiguration step is to swap two tokens on the endpoints of an edge.

Given a reconfiguration graph G_R, the typical reconfiguration problems studied by the community are the following:

1. **Connectivity of** G_R. Can any feasible configuration be reconfigured into any other?

2. **Diameter of G_R.** What is the worst case number of transformations needed to reconfigure one configuration into another?

3. **Reachability of B from A on G_R.** Can a configuration A be reconfigured to configuration B by a sequence of transformations?

4. **Shortest / bounded transformation of A to B on G_R.**

   (a) What is the minimum number of transformations, also known as distance, required to reach configuration B from configuration A? Or, equivalently, phrasing the optimization question as a decision problem: can A be reconfigured to B by using at most k reconfiguration steps?

   (b) What is the actual reconfiguration sequence?

5. **Property optimization.** Given a property p, find a configuration on G_R that optimizes p over all feasible configurations.

Examples of solutions to the above problems for the main topics of this thesis – the triangulation flipping and token swapping – can be found in Sections 2.1 and 3.1.

As already mentioned, the source problems and feasible configurations come in many different forms. Token configurations on graphs are one of the most studied, due to historical reasons – the 15-puzzle and its generalizations have been around since the 19th century – and also because these configurations can represent any problem whose solutions can be described as subsets of vertices [111]. Other frequent configurations include (proper) colourings of graphs, Boolean variable assignments, specific subgraphs of a given graph, and others. We will review some of these and their associated source problems in the following sections 1.2 - 1.5. The configurations can be labelled, depending on whether this makes sense for a particular source problem. In the past, such variants included coloured tokens on graphs, labelled triangulations, labelled subgraph reconfiguration and others [111].

The size of the reconfiguration graph is usually exponential in the size of the source problem input and so it is not possible to construct the graph directly. It is, however, common to assume that testing whether a given configuration is feasible as well as whether two feasible configurations are adjacent in the reconfiguration graph can be done in time polynomial in the original input [137].

The reconfiguration graph is usually undirected since most transformation steps can be reversed. Most commonly the transformation rule involves a single local change, for example, a flip in a triangulation, sliding a single token along an edge, recolouring a single graph vertex, changing a single value of a Boolean variable or reversing the orientation of a single directed edge. In a few variants the transformation rule allows multiple changes in parallel, such as when recolouring graphs by Kempe chains, in parallel token swapping or in simultaneous edge flipping in triangulations.

The reconfiguration problems listed above comprise both structural and algorithmic aspects [111]. Depending on the context, properties of the reconfiguration graph as well as the complexity to compute them have been studied. For example, in the case of graph colourings, there is significant interest in classifying instances for which the reconfiguration graph is connected. In fact, the importance of application to random sampling and counting via Markov chains and rapid mixing has been so prominent that the connectivity problem for k-Colouring has also been known as k-Mixing [137; 111].

The algorithmic aspects focus on determining complexity of the optimization problems listed above. If a problem turns out to be tractable, one aspires to develop exact algorithms and actual reconfiguration sequences. For intractable reconfiguration problems one can develop approximation algorithms or prove inapproximability; and to consider parametrized complexity or particular ‘easier’ classes of instances. A general approach is to mimic the hardness proofs of the source problem or to restrict the input to the reconfiguration problem to simpler instances, similarly as in the study of the source problem. For reconfiguration problems that have been studied across multiple domains, the reconfiguration framework hopes to unify the results and identify general patterns. One example is to specify the conditions under which an intractable [respectively polynomial-time solvable] source problem corresponds to an intractable [respectively polynomial-time solvable] reconfiguration problem.

The rest of this section gives examples of past results within the reconfiguration framework. At present many reconfiguration problems have been explored unevenly and not all of the aspects have been studied in all contexts.

Reachability has been one of the most studied reconfiguration problems in the past and has, in fact, also been called 'the reconfiguration problem' [111]. There are reachability results for variants of graph colouring, satisfiability, independent set, vertex cover, dominating set, matching, shortest path reconfiguration and other problems. Already early on a general pattern suggested that NP-complete [respectively polynomial-time solvable] source problems can correspond to a PSPACE-complete [respectively polynomial-time solvable] reachability reconfiguration problem [85]. This has been confirmed for the Independent Set, Vertex Cover, 4-Colouring, 3-Satisfiability, Dominating Set, Clique and other problems as the NP-complete problems and for the 2-Colouring, 2-Satisfiability, Minimum Spanning Tree and others as the polynomial-time solvable problems [111]. The hardness proofs, moreover, in multiple cases mirror the hardness reductions for the source problems [111]. However, exceptions exist in both directions: for 3-Colouring the source problem is NP-complete and reachability is solvable in polynomial time; for Shortest Path Problem the source problem is solvable in polynomial time and reachability is PSPACE-complete.

For some problems with PSPACE-complete reachability, such as the Independent Set, Vertex Cover, Clique or the Dominating Set problems, 'simpler' classes of graphs (paths, trees, bipartite graphs, etc.) were considered to pinpoint the tractable versus intractable boundary between classes and to compare them to the (in)tractability of the respective classes for the source problems. A general method has been developed to prove PSPACE-completeness of reachability for problems with input constrained to graphs of bounded bandwidth [145]. The method worked for k-Colouring, Shortest Path, Independent Set, Dominating Set and other problems.

Recall the definition of fixed parameter tractability, as given in [111]: a problem is *fixed-parameter tractable (FPT) with respect to parameter* p, if it can be solved in time $f(p)n^{O(1)}$, where n is the input size, p a parameter and f a computable function depending only on p. Moreover, problems intractable with respect to fixed-parameter complexity are $W[\ell]$*-hard*, for $\ell \geq 1$ [111]. Intuitively, parametrized complexity identifies aspects that make a given intractable problem easy/hard. Parametrized complexity has been investigated in the context of Independent Set, Vertex Cover, Dominating Set, List Colouring Reconfiguration and other problems. For reachability as well as shortest transformation it has been common to parametrize the complexity of problems by the length, l, of a reconfiguration sequence. A general method to prove hardness has been devised by using the so-called Subset Problem for a Hereditary Property [108]. This problem is an abstraction of multiple source problems (such as the Independent Set). One version of the general method guarantees that if a reconfiguration graph is formed based on the Subset Problem in a certain way, then reachability parametrized by l is at least as hard as the source Subset Problem. On the other hand, a so-called reconfiguration kernel method has been used to show FPT of reachability based on the FPT of the corresponding source problem for Vertex Cover with TAR adjacency rule parametrized by the solution size, and for other problems [108]. Here TAR denotes the *token-addition-removal* rule, under which at each step a single token is either removed from a graph or added to an empty vertex of the graph. The solution size denotes the maximum number of tokens allowed to be in a configuration at once.

A general way to establish reachability on a reconfiguration graph is to identify a canonical configuration into which every other configuration can be transformed. An example is the Delaunay triangulation in (unlabelled) flip graphs of planar point sets, see Section 2.1. Also in the case of token swapping on a connected graph, it is easy to show that the reconfiguration graph is always connected and any permutation of tokens on graph vertices can be reconfigured into any other. Hence in both domains – in flipping (unlabelled) triangulations and in token swapping on graphs – the main reconfiguration problems of interest have been variants on the shortest transformation problem and the diameter of the reconfiguration graph.

Often a way to provide a lower bound on the length of a shortest transformation for a given pair of configurations, is to quantify how much the two configurations ‘differ’, given the reconfiguration rule. For example, in the case of tokens on graphs one can consider the symmetric difference of the original and target token configuration. For triangulation reconfiguration, the number of edge intersections, when the original and the target triangulation are overlayed on top of each other, gives an upper bound on the number of flips needed to reconfigure one triangulation into the other [71] (note that in contrast to the previous example this result is not trivial to prove).

Shortest transformation for triangulation flipping was shown to be NP-complete and APX-hard [101; 118], but in FPT when parametrized by the length of the reconfiguration sequence [90]. For token swapping on general graphs, the shortest transformation was also proved to be NP-complete and APX-complete [106], and multiple approximation algorithms were developed for general graphs, as well as for trees, see Chapter 3.

Apart from triangulations flipping and token reconfigurations, the shortest transformation problem received a lot of attention in the context of satisfiability, where it was shown that the complexity of shortest transformation (and of reachability) for particular classes of problem instances mirrors the respective results for the source problem [111].

Note that in addition to the five most studied reconfiguration problems that we listed above – connectivity, diameter, reachability, shortest transformation and property optimization on a given reconfiguration graph – there are questions in the reconfiguration framework that focus on classes of reconfiguration graphs. For example, comparing reconfiguration graphs of multiple problems, determining equivalent reconfiguration steps for a given source problem, or determining the minimal solution size so that reachability in token reconfigurations under the TAR adjacency rule becomes possible [111]. Yet other problems in the framework include investigations into the chromatic number, Hamiltonicity, girth and other properties of reconfiguration graphs, see [111].

To conclude, reconfiguration graphs may be too large to be constructed explicitly, but they often seem to provide some smart way of ‘searching in the dark’: for example, proving connectivity leads in multiple contexts to random sampling and counting of the many configurations; in other contexts the local transformation rule together with properties of the configurations provides a way to efficiently reach a particular configuration without the need of an (impossible) exhaustive search, such as, for example, when optimizing properties of a triangulation (see Section 2.3).

In the remainder of this chapter, we survey the reconfiguration results organized by the source problems. Section 1.2 reviews examples of reconfiguration in different fields and their applications. Section 1.3 gives a broader background to reconfiguration of planar graphs. Section 1.4 discusses reconfiguration of matroid bases and, finally, Section 1.5 reconfiguration of tokens on graphs.

1.2 Examples of reconfiguration and applications

Reconfiguration problems appear in numerous, sometimes seemingly unrelated contexts. The aim of this section is to list some of them, and to give a taste of possible applications.

Historically the oldest and possibly the largest class of examples includes reconfiguration of tokens placed on vertices of a graph. We survey the results on token swapping in Chapter 3. Introduction to other types of token reconfiguration, including token sliding, jumping, addition and removal as well as different types of constrained token configurations are covered in Section 1.5.

Another large part of the community studies reconfiguration of planar graphs. See the survey by Bose and Hurtado [30] for a thorough summary. We include a detailed survey of results on reconfiguring triangulations of planar point sets by flips in Chapter 2. Some other types of triangulation reconfiguration are covered in Section 1.3.1. Reconfiguration of pseudo-triangulations, non-crossing spanning trees and perfect matchings are surveyed in Sections 1.3.2, 1.3.3 and 1.3.4, respectively. Generalization of spanning trees to matroids and their reconfiguration is discussed in Section 1.4.

Another prominent class of examples consists of graph recolouring problems. The vertices of the corresponding reconfiguration graph are usually all proper colourings of a given graph G by k colours. The reconfiguration rule specifies how vertices of G can be recoloured. Two frequently used rules involve recolouring a single vertex to any of the k colours, or to a colour from a list of colours assigned to that vertex, subject to the condition that the result of the recolouring is another proper k-colouring of the graph G. Another widely studied rule is to recolour multiple vertices of the graph G at once by a so-called *Kempe chain* (exchange of two colours on a connected component of a subgraph of G that is induced by the vertices of the two colours), again so that the result is a valid k-colouring. The most studied reconfiguration problems include the connectivity of the reconfiguration graph and the complexity of the reachability problem under different conditions. Surveys by van den Heuvel [137] and Nishimura [111] provide summaries on graph recolourings.

Another wide field studies reconfiguration of solutions to the satisfiability problems.

Given a formula with Boolean variables, the vertices of the reconfiguration graph are all variable assignments for which the formula is satisfied. The reconfiguration rule allows to swap one variable to an opposite truth value (0-1) so that the formula stays satisfied. The well-known reconfiguration results in this area in a way mirror the results about the source problem. In particular, the complexity of a satisfiability problem was proved to be in P or NP-complete based on whether the instance is built from the so-called Schaefer relations or not (Schaefer's dichotomy theorem [111]). On the other hand, complexity of the reachability problem was proved to be in P or PSPACE-complete depending on whether the instance is built from so-called tight relations or not. Similar results hold for the diameter problem and for complexities of the connectivity and the shortest transformation problem. The surveys by van den Heuvel [137] and Nishimura [111] give a review.

Yet other popularly studied reconfiguration problems are derived from a variety of games and puzzles (see [73]), or from other graph-theoretic problems, such as the independent set, vertex cover, dominating set, clique, shortest path problem, and others. See [73; 85] or the surveys in [111; 137].

The many applications of reconfiguration problems include optimizing configuration properties, enumeration and random generation of configurations, identifying 'similar' configurations based on their close position in the reconfiguration graph, applications to other fields of mathematics and computer science, to engineering, as well as practical applications to the industry. We mention a few in the remainder of this section.

Regarding optimizing configuration properties, the section on page 48 gives examples of algorithms that use the flip graphs in order to compute triangulations which are optimal with respect to certain quality measures such as the angle sizes or others, out of all possible triangulations of a given point set. See the same section or the discussion in Section 1.3.1 for examples of using reconfiguration and triangular meshes to approximate shapes in computer graphics or to reconstruct surface from samples. Section 2.1 gives examples of applications of triangulation reconfiguration to other fields, such as to the study of associahedra, rotations in binary trees, graph untangling, reconfigurations of other planar graphs, and others. For application of reconfiguration to object animation in computer graphics, or to morphing of graph drawings and of polygons, see the references in [111]. Some of the applications related to reconfiguration of tokens on graphs include string editing algorithms, algorithms for robot motion planning (discussed in Section 1.5.2), for transferring data packets over a network without exceeding capacities of data buffers at individual vertices (see [37]), for controlling memory usage in distributed systems [16] and many others. Graph colouring reconfiguration has been used in assigning radio frequences to transmitters in a continuously changing mobile network so as not to interfere with one another while at the same time using as little frequency range as possible, see [137]. Other versions of the application include routers in a changing network which can be modelled as reconfiguration of independent sets on a graph [89]. Finally, a non-negligible amount of applications of reconfiguration also cover numerous pastime puzzles and games, see the survey by Demaine and Hearn [49].

We conclude the section by describing the two possibly most significant applications of reconfiguration: the enumeration and random sampling of configurations that can be used across a variety of contexts. In their well-known paper, Avis and Fukuda [18] introduced a so-called reverse search paradigm that enables one to enumerate vertices of certain reconfiguration graphs. The idea is to form a spanning tree of the reconfiguration graph based on a so-called local search function that efficiently determines neighbours of each vertex and which has a unique global optimum and, subsequently, to traverse this tree. The paradigm can be used for enumeration of point set triangulations, spanning trees of a given graph, and for many other graph families (but not quite for combinatorial triangulations where a flip can result in an isomorphic triangulation). A good summary is included in the survey by Bose and Hurtado [30].

The other important application is to sample (and, consequently, to enumerate, via a connection described in [87]) configurations almost uniformly at random in polynomial time by performing random walks on the reconfiguration graph. The idea is that the walk starts at an arbitrary vertex of the reconfiguration graph, and at each step it advances to one of its neighbours or stays at the current vertex. After a number of steps, the walk converges to a stationary probability distribution. To be able to efficiently sample the random configurations, one needs to prove, firstly, that the stationary distribution is the uniform distribution and, secondly, that the convergence happens fast, i.e., in polynomial time. More precisely, if Δ is the maximum degree in the reconfiguration graph and $\lambda < 1$ is any positive constant, then one can let the walk advance to each of its k neighbours with probability λ/Δ and stay at the current vertex with probability $1 - k\lambda/\Delta$. Provided that the reconfiguration graph is connected, it has been proven that the corresponding stationary distribution is the uniform random distribution, see the summary in [30]. Regarding the rate of convergence, loosely speaking, if the walk converges arbitrarily closely to the stationary distribution in polynomial time, the walk is said to be *rapidly mixing* (for a rigorous definition of rapid mixing, see, for example [14]). An important role is played by the so-called expander graphs which are known to enable rapid mixing of random walks. These graphs are characterized by a good *edge expansion ratio* which, for a graph G and a subset $S \subseteq V(G)$ of its vertices, is defined as

$$\min_{S \subseteq V(G)} \frac{|E(S, \bar{S})|}{\min\{|S|, |\bar{S}|\}}.$$

Here $E(S, \bar{S})$ is the set of edges with one endpoint in S and another one in its complement $\bar{S} = V(G) \setminus S$.

Random sampling by performing a rapidly mixing random walk on a reconfiguration graph has been used in multiple settings: in graph colouring reconfiguration, for the so-called Glauber dynamics in theoretical physics, see [137]; or to generate a random matroid basis (see Section 1.4); or to generate a random triangulation of a convex polygon [107]. In other settings, where the rapid mixing property has not yet been proved, random walks on reconfiguration graphs can be used as heuristics. The mixing time of a random walk on a flip graph of triangulations of a general planar point set is a major open question.

For more details on random sampling as well as on other applications and examples of reconfiguration, see the surveys [30; 137; 111].

1.3 Reconfiguration of planar graphs

In this setting, vertices of the reconfiguration graph are all members of a particular class of planar graphs. The transformation between them is usually some version of a k-*flip* that consists of removing k edges from the graph followed by inserting k edges so that the graph remains in the same class. There may be bounds on k and additional constraints in order to keep the transformation step small and local.

Class of graphs (feasible configurations)	Reconfiguration step
triangulations	flip
	simultaneous flip
	flip and vertex move
	vertex addition or removal
	edge insertion (k-flip)
pseudo-triangulations	pseudo-flip
	pseudo-flip and edge insertion
spanning trees	edge move
	compatible edge move
	edge rotation
	empty triangle rotation
	edge slide
matchings	compatible with single cycle
	compatible
	compatible disjoint
simple polygons	k-flip
convex subdivisions	k-flip

Table 1.1: Classes of planar graphs and the corresponding transformation steps in the reconfiguration graph. The source problem is to compute an instance of a particular planar graph (for example, a triangulation) on a given set P of n points in the plane. The vertices (feasible configurations) of the reconfiguration graph are then all planar graphs of that class (for example, all triangulations) on P.

A typical example is a class of all triangulations of a given planar point set with a local flip operation (or 1-flip, according to the above definition) that exchanges diagonals of a convex quadrilateral in the triangulation. Other examples include classes of pseudo-triangulations, spanning trees, or perfect matchings, all on a fixed point set. See Table 1.1 for types of planar graphs and transformation steps typically used with them. For most classes, the respective reconfiguration graphs are referred to as *flip graphs*.

The graphs in a particular class can, futhermore, be considered in a combinatorial or geometric setting, and be either labelled or unlabelled. The combinatorial setting does not assume any particular embedding of the graph (and, whenever there is a need to work with a drawing of such a graph, the embedding can use Jordan curves). On the other hand, in the geometric setting graphs are assumed to be embedded in the plane using straight-line, non-crossing edges. The latter, more restrictive setting can make some flips infeasible. For example, in a geometric triangulation a diagonal of a non-convex quadrilateral cannot be flipped since this would introduce edge crossings.

Throughout in the geometric setting we also assume that the given point set is in general position, i.e., it does not contain any three collinear points or four point on a circle.

Variations in which vertices or edges are labelled have been explored especially in the context of triangulations, pseudo-triangulations and spanning trees, see below.

Most results on reconfiguring planar graphs concentrate on the problems of connectivity and diameter. For some classes, such as simple polygons, connectivity of the flip graph when using any constant-size local transformation is still open [30]. On the other hand, for unlabelled triangulations with flips, connectivity of the flip graph has been solved decades ago and the main problem of interest in recent years has been the flip distance and its variations. A comprehensive survey on flips in planar graphs can be found in [30].

1.3.1 Triangulations

As reconfiguration of edge-labelled triangulations is one of the main topics in this thesis, we give a comprehensive overview of the results on triangulation flipping in Chapter 2. In particular, that chapter contains a survey on geometric triangulations of planar point sets and of simple polygons, a summary of results on constrained triangulations and a brief comparison of the geometric and combinatorial triangulations. Further, Chapter 2 also contains a comparison of the classical flip operation to simultaneous flipping and to edge insertion. Finally, that chapter compares results on edge-labelled versus unlabelled geometric triangulations. In this chapter we discuss further extensions of edge flipping and other local transformations.

For reconfiguring geometric triangulations of a point set a local transformation that consists of an edge flip and a point move has been studied. A *point move* modifies the coordinates of a single vertex in a triangulation so long as it does not introduce any edge crossings [1]. Since edge flips can only reconfigure between triangulations of the same point set, the idea was to introduce the point moves to be able to reach any triangulation of any n-vertex point set, similar to the result by Wagner [142] for combinatorial triangulations. A series of results showed that, indeed, for an n-vertex point set and a combination of flips and point moves in the plane, the corresponding flip graph is connected and has diameter $O(n \log n)$ [11]. Alternatively, if the original triangulation is embedded on an $n \times n$ grid and the vertices must stay within a $5n \times 5n$ grid, $O(n^2)$ flips and point moves suffice to transform any triangulation on n points into any other [1], see also [30].

The survey by Bose and Hurtado [30] also discusses triangulations of 2D surfaces other than the plane. If graph embeddings are allowed to use the Jordan curves, it has been shown that any triangulation of n points on a closed surface can be reconfigured into any other by flips, provided that n is large enough [30]. On the other hand, if edges must be embedded as arcs of geodesics, Bose and Hurtado [30] give examples of point sets on a cylinder or torus where one or both diagonals of a convex quadrilateral are exterior to the quadrilateral. The surface thus imposes further restrictions on the feasibility of the flip operation or even leads to instances when a maximal set of edges is not a triangulation. Note that on 2D surfaces triangulations are often required to have all faces, including the outerface, triangular. If a triangulation of a surface has a non-triangular outerface, it may be hard to define the triangulated domain. For example, there is a point set P on a cylinder, such that different triangulations (by arcs) on P result in different triangulated domains. See [30] for details and further references.

In relation to 2D surfaces, Bose and Hurtado also mention piecewise-linear triangulated surfaces embedded in 3-dimensional space such as triangulated polyhedral surfaces homeomorphic to a sphere where all edges are embedded as straight-line non-crossing segments and where a flip alters the surface: it exchanges a pair of triangles, say abc and bcd, for another pair adb and adc that form, in general, a ‘lower’ and ‘upper’ surface of a tetrahedron. A flip is feasible so long as it does not cause any surface self-intersections. A corresponding flip graph may not be connected, nevertheless such reconfigurations are used as heuristics in surface reconstruction from samples of data points [30].

Bern et al. [22] discuss a generalized version of flips in the plane that consists of traditional flips and vertex addition/removal operation. The *vertex addition* inserts a vertex into an interior face of a triangulation and connects it to all three vertices of the face. The *vertex removal* does the reverse. The two types of operations – flips and vertex addition/removal – correspond to two different projections of a tetrahedron into 2D: one, in which the four faces of a tetrahedron project onto two triangles, and one in

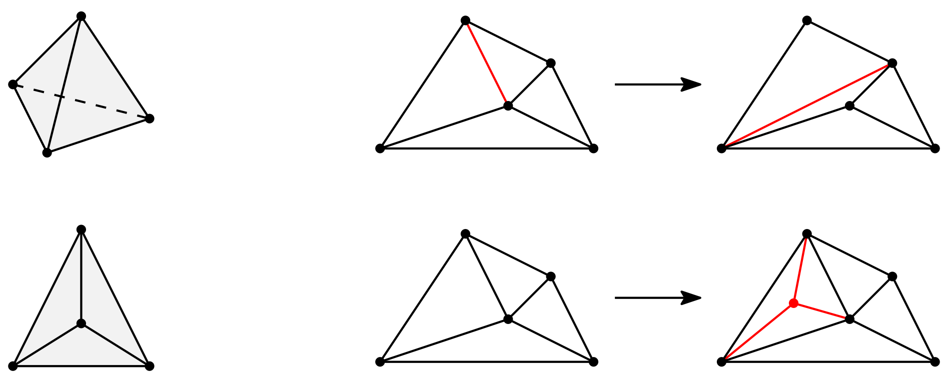


Figure 1.2: Left: Two projections of a tetrahedron to a plane, viewed from above. Right: Generalized 2D flips derived from the projections on the left. The flips can be generalized to higher dimensions and to cubical meshes.

which they project onto a single triangle. Both operations can then be viewed as 'flipping' between the upper and lower triangles of the tetrahedron in the projection, see Figure 1.2 (a similar figure appeared in [22]). These operations generalize to higher dimensions: for tetrahedralizations, different types of flips correspond to different ways of cutting the surface of a 4-dimensional simplex into a 'lower and upper half', and so on. Bern et al. show that the method can be extended to cubical meshes. Instead of triangular faces (or simplices), the cubical meshes use quadrilateral faces (or hypercubes in higher dimensions). Similarly as for triangulations, flips in d-dimensional cubical meshes can also be defined by using a $(d+1)$-dimensional hypercube [22].

Turning to labelled triangulations in the plane, Chapter 2 reviews and gives new results in the context of edge-labelled geometric triangulations. Vertex-labelled combinatorial triangulations are also discussed along the way in Section 2.3.

Reconfiguration by using edge flips and point moves, as described above, also works with vertex-labelled geometric triangulations. Abellanas et al. [1] proved that $O(n^2)$ edge flips and point moves, where again the sizes of vertex coordinates are bounded, are sufficient to reconfigure any labelled triangulation of an n-vertex point set into any other.

Yet other examples of reconfiguring edge-labelled geometric triangulations are described by Eppstein in [58]. Eppstein studies similarities between three different types of edge-labellings. Here we discuss two of them, namely Schnyder and rectangular labellings. Note that the respective reconfiguration graphs have as feasible configurations all possible labellings of a single triangulation, rather than (all labellings of) all triangulations of a given point set. In both labelling types, the triangulation edges are assigned colours and orientations so as to satisfy certain local conditions. Each such labelling can be constructed efficiently in linear time.Schnyder labellings are defined only for triangulations with a triangular outerface, while rectangular labellings use triangulations with a quadrilateral outerface and no separating triangle – these restrictions make sense based on the context in which the labellings are used, see below.

Schnyder labellings were used to construct an efficient algorithm to embed a maximal planar graph on n vertices in the plane with straight-line edges and vertices as nodes of the integer grid of linear dimension [58]. Two Schnyder labellings can be related by a (non-reversible) *twisting operation*. This produces a reconfiguration graph which is directed and has been proven to be acyclic [58].

The so-called rectangular labellings of triangulations encode partitions of a rectangular region into smaller rectangles, hence the name. The labellings enable one to generate the corresponding partitions efficiently. One such labelling can be transformed into another one again by a certain *twisting operation* that involves switching colours and orientations of some edges in the labelling locally. In this case all rectangular labellings of the same triangulation can be reached by a sequence of twisting operations, so the corresponding flip graph is connected [58].

1.3.2 Pseudo-triangulations

A *pseudo-triangle* is a simple polygon that has exactly three interior angles convex. A *pseudo-triangulation* of an n-vertex point set P in the plane is a subgraph of a triangulation that contains all convex hull edges of P and whose interior faces are all pseudo-triangles with empty interior. A pseudo-triangulation is called *pointed* if every of its vertices has one of its incident angles reflex. It can be shown that every pointed pseudo-triangulation on n vertices has exactly $2n - 3$ edges and $n - 1$ faces. This is also the minimum number of edges that a pseudo-triangulation can have, as proved by Streinu [134].

An interior edge e in a pseudo-triangulation is incident to two pseudo-triangles T_1 and T_2. Suppose that these have their three convex angles at vertices a, b, c and x, y, z, respectively. Further assume that e is on the edge chain joining a to b in T_1 and x to

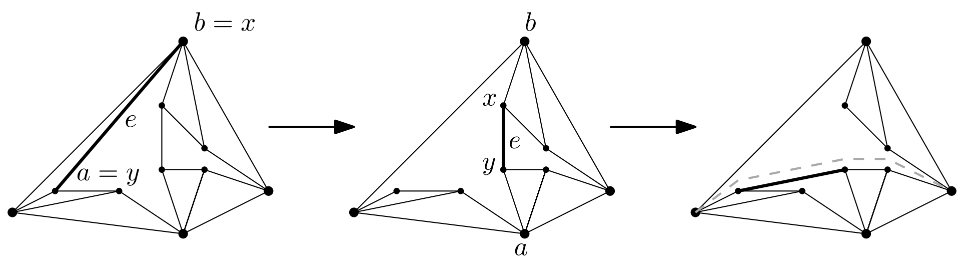


Figure 1.3: Examples of two pseudo-flips. The first one results in a deletion of an edge. In each pseudo-flip, we label vertices a, b, x, y.

y in T_2. Then a *pseudo-flip* of e removes e from the pseudo-triangulation and inserts all edges on a geodesic within the created pseudoquadrilateral joining c and z that were not yet part of the pseudo-triangulation. The result of this operation is another pseudo-triangulation. Note that sometimes a pseudo-flip consists of a single deletion of an edge, but it can be shown that in the case of pointed pseudo-triangulations every pseudo-flip removes and inserts exactly one edge [34; 134]. See Figure 1.3 for an example of pseudo-flips.

By using a canonical pointed pseudo-triangulation, Bereg [20] proved that the flip graph of pointed pseudo-triangulations with pseudo-flips is connected and has diameter $O(n \log n)$. For general pseudo-triangulations of a point set the corresponding flip graph is connected if, in addition to pseudo-flips, one also allows insertion of a single edge as an operation, as long as it produces a pseudo-triangulation. In this case the diameter of the flip graph is also $O(n \log n)$, proven by Aichholzer et al. [5].

Reconfiguring edge-labelled pseudo-triangulations has been considered in [33]. The authors proved that $O(n^2)$ pseudo-flips are always sufficient to transform any edge-labelled pointed pseudo-triangulation into any other (thus the situation is quite different from edge-labelled geometric triangulations where the flip graph can be disconnected, see Section 2.5).

1.3.3 Non-crossing spanning trees

Spanning trees and their reconfigurations have been researched in many variations: in the combinatorial and geometric setting, as labelled or unlabelled graphs, using various reconfiguration steps, and for flips performed one after another or simultaneously.

In the combinatorial setting, spanning trees on n (labelled or unlabelled) vertices are studied as abstract graphs. There are n^{n-2} different abstract spanning trees if the points are labelled [41]. For unlabelled points, the vertices of the reconfiguration graph are the isomorphism classes of trees.

Probably the most natural reconfiguration step on an n-vertex spanning tree T is an *edge move* that deletes an edge uv and inserts another edge so that the graph remains a spanning tree. Another, more restrictive operation is an *edge rotation*, for which the deleted and the inserted edge must have a common endvertex u. A third possible operation is an *edge slide*. This is a rotation inserting an edge uw such that vertices v and w are adjacent in the original tree T.

The reconfiguration graphs formed in the combinatorial setting for labelled spanning trees using the edge move operation or for unlabelled spanning trees using any of the above three operations, have been proved to be connected and with a linear diameter. See [110] for more details on the combinatorial setting and references to original results.

In the geometric setting, one considers the set $\mathcal{T}_P$ of spanning trees with pairwise non-crossing straight-line edges on a fixed set P of n points in the plane. Known bounds on the size of $\mathcal{T}_P$ are $O(141.07^n)$ [77] and $\Omega(12.54^n)$ [80]. The operations of edge move, rotation and slide work similarly as in the combinatorial setting with the additional constraint that in order to be feasible, they must output a non-crossing spanning tree. In the case of an edge slide uv to uw, where v and w are adjacent, we additionally require that the triangle uvw is empty. In the geometric setting more reconfiguration operations were considered. These include an *improving edge move*, a *compatible edge move* and an *empty-triangle rotation* [110]. The *improving edge move* is an edge move that additionally decreases the Euclidean length of the spanning tree. A *compatible edge move* is a move in which the deleted and the inserted edge do not cross. Finally, an *empty-triangle rotation* is a rotation of edge uv into uw such that the triangle uvw has an empty interior. All operations apart from the improving edge move are reversible and produce an undirected flip graph. Note also that for the set $\mathcal{T}_P$ the operations satisfy the following inclusions:

$$\begin{aligned}\text{edge slides} \subseteq \text{ empty-triangle rotations} \subseteq \text{ rotations} \subseteq \\ \subseteq \text{ compatible edge moves} \subseteq \text{ edge moves}\end{aligned}$$

and so also the corresponding flip graphs are subgraphs of each other.

In the context of enumerating the set $\mathcal{T}_P$ via a reverse search algorithm, Avis and Fukuda [18] proved the connectivity of, and provided the first upper bound, $2n - 4$, on the diameter of the flip graph for $\mathcal{T}_P$ with edge rotations (as explained in [110], the proof was phrased in terms of edge moves, but all used moves were actually rotations). What followed were results on the special case when the point set is in convex position: the flip graph for edge moves is Hamiltonian [76], has a minimum number of vertices as compared to reconfiguration graphs of spanning trees on the same-size but non-convex point sets [66], and there is a lower bound of $3n/2 - 5$ on its diameter [76]. This lower bound is currently the best known for a general point set and all of the above-mentioned operations apart from the edge slides. The survey [110] lists known diameter bounds for different scenarios in comprehensive tables.

Aichholzer et al. [4] proved the Fixed Tree Theorem. Given a spanning tree T on a set P of n points, they study sequences of length-decreasing trees $T, f(T), f^2(T), \ldots$, where f assigns to T a minimum length spanning tree $f(T)$ that is compatible with T, i.e., such that the union of edges in $f(T)$ and T form a plane graph. The Fixed Tree Theorem states that $f(T) = T$ if and only if T is the minimum length spanning tree on P. Moreover, starting with any tree $T \in \mathcal{T}_P$, the sequence reaches the minimum spanning tree of P in $\Theta(\log n)$ steps. The function f is an example of a simultaneous compatible edge move on the tree T, and $O(\log n)$ is the diameter of the flip graph formed on $\mathcal{T}_P$ by the simultaneous compatible edge moves. Aichholzer et al. [4] also showed that each application of f can be substituted by $O(n)$ improving edge moves. Thus the directed flip graph of $\mathcal{T}_P$ with improving edge moves is weakly connected and has diameter $O(n \log n)$. The minimum spanning tree of P is a unique sink in the flip graph, similarly as the Delaunay triangulation is for the Delaunay flips in the flip graph of triangulations (see Sections 2.2 and 2.3).

Aichholzer and Reinhardt [8] also proved that any spanning tree in $\mathcal{T}_P$ can be reconfigured to any other by $O(n^2)$ edge slides, thus showing that in fact the flip graphs of $\mathcal{T}_P$ are connected for all the above-mentioned reconfiguration operations. Perhaps even more importantly, the edge slides demonstrate that reconfiguration of spanning trees is possible by constant-size and local transformation that can even be performed in a continuous manner [4].

Finally, reconfiguring edge-labelled spanning trees by edge moves that assign the deleted edge label to the newly inserted edge also produces a connected flip graph [75].

For more results, including simultaneous flipping, see the recent survey [110].

1.3.4 Perfect Matchings

Another example of planar graphs whose reconfiguration has been studied are perfect matchings. The combinatorial setting in this case turns out to be trivial, hence all results in this section assume the geometric setting. The set of vertices of a reconfiguration graph is usually the set $\mathcal{M}_P$ of all non-crossing perfect matchings on a given point set P of $2n$ points in the plane. A *perfect matching* on P is a set of n straight-line segments so that each point in P is a vertex of degree one. A perfect matching is said to be *non-crossing* if none of its edges pairwise intersect.

If the $2n$ points of P are in convex position, the number of different non-crossing perfect matchings on P is known to equal the nth Catalan number, $\frac{1}{n+1}\binom{2n}{n}$. It can be shown that this is the smallest number of non-crossing perfect matchings that a $2n$-vertex point set can have [66]. There also are bounds on the maximum size of $\mathcal{M}_P$ for a general point set P, though not tight. See the survey in [3] for details and further references.

A symmetric difference of two matchings M_1 and M_2 from $\mathcal{M}_P$ is in general a number of possibly (self-)intersecting cycles of alternating edges from M_1 and M_2. Three types of adjacency in the reconfiguration graph have been defined based on this symmetric difference: adjacency by matchings being *compatible*, *compatible with single cycle*, or *disjoint compatible*. Two matchings in $\mathcal{M}_P$ are called *compatible* if their symmetric difference is a plane graph. They are *compatible with single cycle* if, additionally, the symmetric difference consists of a single alternating cycle. Finally, they are *disjoint compatible* if they are compatible and have no edge in common. All these three types of adjacency are examples of k-flips, since a single reconfiguration step exchanges up to n edges in the matching.

Houle et al. [78] proved that the reconfiguration graph of non-crossing perfect matchings using the compatible-with-single-cycle adjacency is connected and has diameter $2n - 2$. Connectivity of the graph is, however, open if the number k of edges flipped in a single step is required to be bounded. When reconfiguration is done exclusively by 2-flips (i.e. when the symmetric difference must be a single cycle of 4 alternating edges), the authors of [78] report that the corresponding flip graph has no isolated vertices (but the proof is not included).

Hernando et al. [74] solved the above case of 2-flips for point sets in convex position. The corresponding flip graph was proved to be connected, with diameter $n - 1$, bipartite for every n, having minimum degree $n - 1$; and containing a Hamiltonian cycle if and only if $n \geq 4$ is even.

The result in Houle et al. [78] implies that the reconfiguration graph is also connected when adjacent matchings are just compatible, dropping the requirement that they must be compatible with single cycle. In this case, Aichholzer et al. [6] and Razen [124] proved the upper and lower bounds on its diameter, $O(\log n)$ and $\Omega(\log n / \log \log n)$, respectively.

Aichholzer et al. [6] showed that a non-crossing perfect matching on $2n$ points in the plane, where n is odd, may not have any disjoint compatible matching and, hence, the corresponding flip graph contains an isolated vertex. The opposite is true when n is even: the Disjoint Compatible Matching Theorem proved in [84] guarantees that for any non-crossing perfect matching on $4m$ points there is another disjoint compatible non-crosing perfect matching. In general, however, the reconfiguration graph in which adjacency is defined by disjoint compatibility is disconnected. Aichholzer et al. [3] studied the case for the special case when the point set is in convex position. They proved that the disjoint compatibility flip graph is disconnected for $n \geq 3$ and characterized its connected components.

Yet another version of a matching reconfiguration problem is the study of *bichromatic matchings*. These are non-crossing perfect matchings on n blue and n red points in the plane in which every edge connects vertices of different colours. The reconfiguration graph of bichromatic matchings where two matchings are adjacent if they are compatible (but not necessarily disjoint or single cycle) is connected. See [10] for details of the proof as well as for further background on bichromatic matchings.

Finally, Biniaz et al. [25] consider a different but related problem of flipping perfect matchings, namely, the transformations by which a given perfect matching can be turned into a non-crossing perfect matching on the same point set. Their operation is a 2-flip that 'uncrosses' a pair of edges. The problem can be thought of as finding a shortest path in the reconfiguration graph whose vertices are all perfect matchings on the given point set to a vertex in a subgraph consisting of all non-crossing perfect matchings.

1.3.5 Other planar graphs

Apart from triangulations, pseudo-triangulations, spanning trees and perfect matchings that were described in the previous sections, reconfigurations of other types of planar graphs by means of k-flips have been considered. Simple polygons (also known as polygonizations or non-crossing Hamiltonian cycles) received considerable attention because their reconfiguration could be used to generate a random simple polygon on the underlying point set [30]. Other examples include reconfiguration of Hamiltonian paths or of convex subdivisions of a planar point set. The latter are generalizations of triangulations in which every face is a convex polygon. Similarly as for perfect matchings, the main open problem in many of these contexts is to establish local constant-size transformations that result in a connected flip graph, see [30].

1.4 Reconfiguration of matroid bases

Reconfiguration of matroids has been discussed by Ito et al. [85], by Anari et al. [14] and by Lubiw and Pathak [102].

Recall that a *matroid* $\mathcal{M}$ is a pair $(\mathcal{E}, \mathcal{I})$ where $\mathcal{E}$ is a ground set and $\mathcal{I}$ is a collection of subsets of $\mathcal{E}$, called the *independent sets*, such that $\mathcal{I}$ satisfies the following properties:

1. $\emptyset \in \mathcal{I}$,
2. hereditary property: if $A \in \mathcal{I}$ and $B \subseteq A$ then $B \in \mathcal{I}$,
3. augmentation property: if $A, B \in \mathcal{I}$ and $|A| > |B|$ then there is an element $a \in A \setminus B$ such that $B \cup \{a\} \in \mathcal{I}$.

Any set of elements of $\mathcal{E}$ that is not independent is called *dependent*. A set $A \in \mathcal{I}$ is called a *basis* if it is a maximal independent set, i.e., adding any other element of $\mathcal{E}$ to A would make it dependent. Property 3 implies that all bases of $\mathcal{M}$ have the same size, called the *rank* $r(\mathcal{M})$ of the matroid. A matroid can equivalently be defined in terms of its set of bases or in terms of its set of *circuits*, which are the minimal dependent sets of elements of $\mathcal{E}$.

Matroids generalize the notion of linear independence and bases of vector spaces from linear algebra and also the notion of independence in graph theory. Matroids that are derived from graphs are called graphic. In that case $\mathcal{E}$ consists of all edges of a given connected graph G, the independent sets are exactly the cycle-free subgraphs of G and the bases correspond to edge sets of all spanning trees of G. For an introduction to matroids and their properties, see the textbook by Oxley [115].

A fundamental property of matroids is the so-called *basis exchange*: if A, B are two distinct bases of a matroid $\mathcal{M}$ and $a \in A \setminus B$, then there is an element $b \in B \setminus A$ such that $(A \setminus \{a\}) \cup \{b\}$ is a basis of $\mathcal{M}$. See [115] for a proof.

Ito et al. [85] used a weighted version of the basis exchange property to show that, given a connected graph G with a non-negative weight on each edge, reconfiguring one spanning tree of G with weight $< k$ into another such spanning tree, using the edge exchange operation and going only through intermediate spanning trees of weight $< k$ is always possible.

More generally, consider the reconfiguration graph whose vertices are bases of a given matroid $\mathcal{M}$ and two vertices are adjacent if the corresponding bases differ by a single basis exchange step. Then, by the basis exchange property, the reconfiguration graph is connected and, for any pair of bases A, B of $\mathcal{M}$, A can be transformed into B in $|A - B|$ steps. Consider the following random walk on the reconfiguration graph: the walk starts at an arbitrary basis A_1 of $\mathcal{M}$ and, at a step i, forms basis A_{i+1} from A_i as follows. One element a of the basis A_i is chosen uniformly at random and is deleted and, out of all elements of the matroid that can complete $A_i \setminus \{a\}$ to a basis, one element, say b, is chosen uniformly at random and $A_{i+1} := \{A_i \setminus \{a\}\} \cup \{b\}$. It is easy to see that for any pair of bases A, B and a step i, the probability of going from a basis A to B is the same as going from B to A, and hence, the stationary distribution of such a walk is the uniform distribution over all bases of $\mathcal{M}$.

In a recent breakthrough, Anari et al. [14] proved that the above walk mixes rapidly, that is, the above walk converges arbitrarily closely to the uniform distribution in polynomial time (for a rigorous definition of rapid mixing, see [14]). The above random walk on the reconfiguration graph thus provides the first polynomial-time algorithm to approximately generate a matroid basis uniformly at random in an arbitrary matroid. In graphic matroids, this enables one to generate random spanning forests of a given graph. Using the equivalence between being able to approximately sample and to approximately count in so-called self-reducible problems [87], the above result implies that it is also possible to approximately count the number of bases of an arbitrary matroid in polynomial time. Another consequence of the above result is that the expansion ratio of the bases exchange reconfiguration graph is at least one, as was conjectured thirty years ago by Mihail and Vazirani [105]. Recall that the expansion ratio of a graph G is defined as

$$\min_{S \subseteq V(G)} \frac{|E(S,\bar{S})|}{\min\{|S|,|\bar{S}|\}},$$

where S is a subset of vertices of G, $\bar{S}$ is its complement and $|E(S,\bar{S})|$ denotes the number of edges between S and $\bar{S}$. As a sideremark, also note that the rapid mixing result by Anari et al. [14] is not specific to matroids, and holds for any probability distribution that is *d-homogeneous* and *strongly log-concave*, as defined in [14].

Lubiw and Pathak [102] characterized when a labelled matroid basis can be reconfigured into another labelled basis and proved bounds on the diameter of connected components of the corresponding reconfiguration graph. The vertices of the graph are all labelled bases (A,ℓ) of a given matroid $\mathcal{M}$, where A is a basis of rank r and $\ell : A \to \{1,2,\ldots,r\}$ is a one-to-one labelling function. Two labelled bases are adjacent in the reconfiguration graph if one can be obtained from the other by a single basis exchange step, where a basis exchange assigns the label of the deleted element to the newly added element. Lubiw and Pathak proved that a labelled basis (A_1,ℓ_1) can be reconfigured into a basis (A_2,ℓ_2) if and only if, for each label l, the elements in A_1 and A_2 having label l belong to the same connected component of the matroid. Two elements e and f of a matroid are said to be in the same connected component if $e = f$ or if there exists a circuit of $\mathcal{M}$ that contains both e and f. They proved a bound $O(r^{1.5})$ on the diamater of connected components of the reconfiguration graph. In the case of graphic matroids, they improved the bound to $O(r \log r)$.

1.5 Reconfiguration of token arrangements

Ample results on reconfiguration come from the context of token arrangements. Here the feasible configurations (i.e. the vertices of the reconfiguration graph) are different assignments of tokens to vertices of a given graph and a reconfiguration step rearranges a subset of the tokens.

The area grew out of the study of the 15-Puzzle (for puzzle's description, see the very introduction to Chapter 1) and of its generalizations, and nowadays contains a rich collection of problem variants. Tokens on graphs also serve as an abstraction for practical problems, such as robot motion planning (see Section 1.5.2) and, as remarked in [111], they can in general represent any problem whose solution can be described as a subset of vertices on a graph (for example, also reconfiguration of perfect matchings or of spanning trees discussed in previous sections can via line graphs be phrased in terms of tokens on graphs [137]).

Due to the breadth of the field we will only concentrate here on results broadly connected to token swapping and do not attempt a complete survey. An interested reader can consult the surveys in [137; 111]. For pebbling games played on directed graphs and their relevance to complexity theory, see the survey by Nordström [113].

Token reconfiguration problems generally assume a given undirected graph G. To define feasible configurations, one specifies the number of tokens to be placed on G's vertices, their types and allowed arrangements. Usually each vertex of G can hold at most one token at a time. The tokens can be all distinct (*labelled*), or all indistinguishable (*unlabelled*), or in general, *coloured*, where different colours denote different tokens but tokens within the same colour class are indistinguishable. Additionally, the problem may require that only certain token arrangements on G are considered feasible, for example, requiring that the vertices covered with tokens form an independent set in G.

Common reconfiguration rules include *swapping* that exchanges a pair of tokens placed on adjacent vertices of G, or *sliding* that slides a token along an edge to an empty adjacent vertex. A *jump* (also known as a *move*) moves a token to any empty vertex on the graph. A *rotation* picks a simple cycle in G and pushes all tokens on its vertices by one vertex around that cycle.

Feasible configuration			Reconfig. rule	Tackled reconfiguration problems
Number of tokens k	Token types	Token arrange-ment		
1.5.1 Token Swapping and Sliding				
$k = n$	labelled coloured	any	swap	see Chapter 3
$k = n - 1$	labelled	any	slide	connectivity, reachability (Wilson [144]) shortest transformation (Goldreich [67], Ratner and Warmuth [123])
	coloured			shortest transformation (Yamanaka et al. [147])
any $k < n$	labelled			reachability (Kornhauser et al. [95], Auletta et al. [16]) diameter (Kornhauser et al. [95])
	coloured			reachability (Goraly and Hassin [68])
	unlabelled			reachability (easy) shortest transformation (Călinescu et al. [37]) diameter and other graph properties (Fabila-Monroy et al. [61])

Table 1.2: Some studied variants of token reconfiguration problems, part I. Here n is the number of vertices in the input graph G.

Feasible configuration			Reconfig. rule	Tackled reconfiguration problems
Number of tokens k	Token types	Token arrangement		
1.5.2 Other sliding variants and motion planning				
$k < n$	labelled unlabelled	any	empty path slide	shortest transformation (Călinescu et al. [37])
$k < n$	unlabelled with one labelled robot	any & final position of obstacles does not matter	slide	shortest transformation (Papadimitriou et al. [117], Auletta and Persiano [17])
$k = n$	labelled	any	robot carrying tokens	Graf [69]
$k \leq n$	labelled	any	rotation & slides	connectivity, reachability (Yu and Rus [155]) shortest transformation (Yu [153], Yu and LaValle, [154])
$k = n$				connectivity, reachability, diameter (Foerster et al. [62])
$k = n$			rotation around base cycles	connectivity, reachability (Scherphius [125], Yang [151])
1.5.3 Constrained token configurations				
$k < n$	unlabelled	indep. set	slide	reachability (Hearn and Demaine [73])
			TAR / jump / multiple token jump	reachability (Ito et al. [85], De Berg et al. [47]) equivalence of TAR and jump (Kamiński et al. [89])

Table 1.3: Some studied variants of token reconfiguration problems, part II. Here n is the number of vertices in the input graph G.

Tables 1.2 and 1.3 list common combinations of parameter choices – defining the feasible configurations, the transformation rule and, hence, the reconfiguration graph – together with the reconfiguration problems studied in the literature. Generally, research in token reconfiguration has focused on the problems of connectivity, reachability and shortest transformation (see Section 1.1) in the respective reconfiguration graphs.

The next subsections discuss results on token swapping and sliding, on broader variants of sliding, and on constrained token configurations.

1.5.1 Token swapping and sliding

A detailed survey on token swapping on graphs is included separately in Chapter 3, as token swapping is one of the main topics in this thesis. We will concentrate there mainly on distance questions, but also mention some previous work on deciding reachability and on diameter. Here we only remark that the shortest transformation problem for a given pair of labelled/coloured tokens on a graph has also been investigated by using parallel swaps performed on non-adjacent edges [92].

A related operation to swapping is token sliding. In fact, all sliding puzzles can be thought of as containing transparent 'hole' tokens and performing a special type of swaps in which one of the swapped tokens must be a 'hole'.

The classic 15-Puzzle was already studied in the 19th century, see [49; 137; 111] for surveys. It is a version of token sliding on a 4×4 grid graph with 15 labelled tokens and a hole. As a consequence, only half the token configurations (the alternating group) can be reached. Generalizing beyond the 4×4 grid to general graphs, Wilson [144] in 1974 gave a complete characterization of which token configurations on which graphs can be reached via token sliding, hence deciding connectivity or reachability on the corresponding reconfiguration graph can be done in polynomial time. Minimizing the number of token slides is NP-complete [67] even for grid graphs [123]. Recently, a version with $n-1$ coloured tokens on n-vertex graphs has also been considered [147] (note that the problem is phrased in terms of n tokens being swapped, but each swap involves the same particular token). They show it is APX-hard to minimize number of moves, but polynomial time for trees, complete graphs and cycles.

Several papers explore generalizations of the 15-Puzzle to fewer tokens. For $k < n$ labelled tokens on an n-vertex graph, Kornhauser et al. [95] in 1984 gave a polynomial time algorithm to decide if reconfiguration between two token placements is possible, and proved a tight bound of $O(n^3)$ on the diameter of the associated reconfiguration graph. For $k < n$ labelled tokens on a tree Auletta et al. [16] gave a linear time algorithm to decide reachability between two token configurations. This result was used to get a linear time algorithm for reachability for $k < n$ coloured tokens on any graph by Goraly and Hassin [68].

For $k \leq n$ unlabelled sliding tokens, many of the reconfiguration problems become easy, as pointed out in [37; 137]. Two token configurations on an n-vertex graph G are reachable from each other if and only if each connected component of G contains the same number of tokens in both configuration. The minimum number of token slides can be found in polynomial time and the diameter of the reconfiguration graph is at most n^2 [37]. Fabila-Monroy et al. [61] showed that the diameter of the reconfiguration graph (which they call the "token graph") is at most k times the diameter of the original graph. The authors also study other graph-theoretic properties of the reconfiguration graph like connectivity, chromatic number, Hamiltonian paths, and others.

1.5.2 Other sliding variants and motion planning

Călinescu et al. [37] consider a different notion of the distance where a token in one move can slide through a path of empty vertices. They show that computing the minimum number of moves is APX-hard even for unlabelled tokens.

Papadimitriou et al. [117] considered a "motion planning" version where all the $k < n$ tokens are unlabelled "obstacles" except for one labelled "robot" token. The goal is to move the robot from a start vertex to a destination vertex by sliding the robot or the obstacles. Note that the endpositions of the obstacles do not matter in this variant. They showed that minimizing the number of moves is NP-complete for planar graphs but solvable in polynomial time for trees. The run time for trees was improved in [17].

In another variant, there are $k = n$ labelled tokens and token movement must be carried out by a single robot walking along the graph edges and carrying at most one token at a time. The robot does not count as a token and when carrying a token, the robot can pass through other vertices containing tokens until he decides to exchange the token for another one at some vertex. Graf [69] includes a good summary.

Rather than token sliding across an edge, an alternative motion is rotation of tokens around a simple cycle in the graph. At one step the tokens can rotate around multiple disjoint cycles or slide to an adjacent empty vertex. This is of interest in the robotics community since it models movement of robots with the restriction that no two robots can travel along the same edge at the same time. When all cycles in a graph may be used, there are polynomial time algorithms to decide if reconfiguration is possible [155; 62]. Diameter of the reconfiguration graph is also polynomial [155; 62]. See also [153] for hardness of shortest transformation (with several ways to measure distance) and [154] for practical approaches. If rotation is only allowed around the cycles of a cycle basis (e.g., the faces of a planar graph) Scherphuis [125] provided a characterization (similar to Wilson's for the 15-puzzle generalization) of which graph/cycle-basis/token-placement combinations permit reconfiguration (see also Yang [151], who relates Scherphuis' result to Wilson's result).

1.5.3 Constrained token configurations

Sometimes tokens placed on a graph must satisfy some constraints in order to be considered a feasible configuration. For example, the vertices occupied by tokens may be required to form an independent set in the underlying graph, i.e. no edge can connect two vertices with a token. Hearn and Demaine [73] showed that deciding whether two independent set configurations of k unlabelled tokens on an n-vertex graph can be reconfigured into each other by sliding is PSPACE-complete, even when restricted to planar graphs with maximum degree three (in fact, the result holds even for deciding whether a particular token in the independent set configuration can be moved). The paper proves the PSPACE-completness of reachability for multiple other reconfiguration puzzles, not necessarily involving tokens.

Reconfiguration of an independent set of k tokens on a graph has also been studied using different reconfiguration rules, in particular token jumping; and token addition and removal (TAR). Note that it is always possible to reconfigure one independent set of tokens into another if one is allowed to remove sufficiently many tokens. Thus when using TAR, the input to a decision problem also specifies an integer r such that in the reconfiguration sequence every intermediate token configuration must contain at least $k - r$ tokens. Ito et al. [85] proved that the reachability problem for independent set reconfiguration with TAR is PSPACE-complete, again, even when restricted to planar graphs with maximum degree three. Kamiński et al. [89] showed that using token jumping on independent sets is equivalent to using TAR with $r = 1$, i.e. two independent set configurations are reconfigurable from one another by using token jumping if and only if they are reconfigurable by using TAR with $r = 1$. Intuitively, this is because any reconfiguration sequence using TAR can be reordered so that token removals and additions alternate, translating it thus to a (half as long) token jumping sequence. De Berg et al. [47] study the minimum value of r for which reconfiguration of two independent sets via TAR is possible and the structural parameters of the underlying graph influencing it. They also introduce *multiple token jumping*, a reconfiguration step when r tokens can make a jump at once, and again study the minimum value of r for which the reconfiguration is feasible.

Finally, let us note that similarly as for independent sets, reconfiguration of other token arangements has been studied as well, for example, reconfiguration of cliques and vertex covers, see e.g. [85].

2 Triangulation Reconfiguration and a Proof of the Orbit Conjecture for Edge-Labelled Triangulations

In this chapter we discuss problems related to reconfiguration of triangulations and present new results on flipping edge-labelled triangulations. Our main result is a proof of the Orbit Conjecture for edge-labelled triangulations, which characterizes when one edge-labelled triangulation can be reconfigured to another using edge flips.

We start in Section 2.1 by providing basic definitions and introducing the reconfiguration problems in the context of triangulations. Section 2.2 summarizes the main triangulation properties and Section 2.3 gives a survey of classical results on triangulation reconfiguration and flip graphs, including flip graphs of triangulations of planar point sets, simple polygons, convex point set and of combinatorial triangulations; we also briefly review simultaneous flipping and edge insertion as an alternative way to reconfigure.

Sections 2.4 to 2.11 cover labelled triangulations. After a general introduction, we give the basic definitions and motivations for edge labelling in Section 2.5. The Orbit Theorem as our main result is stated in Section 2.6, together with an overview of related results. The proof of the Orbit Theorem occupies Sections 2.7 – 2.10. We next extend the proof of the Orbit Theorem to cover constrained edge-labelled triangulations, which is done in Section 2.11.

We conclude the chapter in Section 2.12 by showing that a shortest flip sequence reconfiguring one triangulation into another may need to flip edges that already have the correct position and label, thus bridging into the next chapter where we will explore analogous problems in the realm of token swapping.

2.1 Basic definitions and reconfiguration set-up

Given a set P of n points in a plane, a *triangulation of* P is a maximal set T of pairwise non-crossing line segments, also called edges, whose end vertices are the points of P. When triangulating a simple, not necessarily convex polygon with vertex set Q, the edges of the triangulation are additionally required to be internal diagonals of the polygon.

Throughout we assume that the points in P and Q are in general position, i.e., we require that no three points lie on a line and no four points on a circle. We concentrate on geometric planar straight-edge triangulations, however, at times we do review some results on combinatorial triangulations for comparison.

Triangulations have traditionally been studied as reconfiguration problems. A fundamental role is played by the flip operation that transforms one triangulation into another and by the reconfiguration graph that is in this context known as the flip graph. More precisely, an edge e of a triangulation T is called *flippable* if the two triangles incident to e in T form a convex quadrilateral. Such an edge can be flipped, where a *flip* replaces the edge e with the opposite diagonal of the quadrilateral and thus produces another triangulation T'. The *flip graph* has a vertex for each triangulation of the given point set (or polygon), and an edge whenever two triangulations differ by one flip. Flips are reversible, hence flip graphs are undirected. An example of a flip graph is given in Figure 2.1.

The following is a list of classic reconfiguration problems that have been studied in the context of triangulations. We cover the details in Section 2.3.

Flip graph connectivity. For the classic cases of geometric triangulations of point sets and polygons, as well as for combinatorial triangulations, the flip graph is connected [97; 98; 52; 142]. This means that any triangulation can be flipped to any other. For the edge-labelled triangulations that we define in Section 2.5, the flip graph may be disconnected.

Diameter of the flip graph. The *diameter* of the flip graph gives the worst-case number of flips required to reconfigure one triangulation to another. Even though the

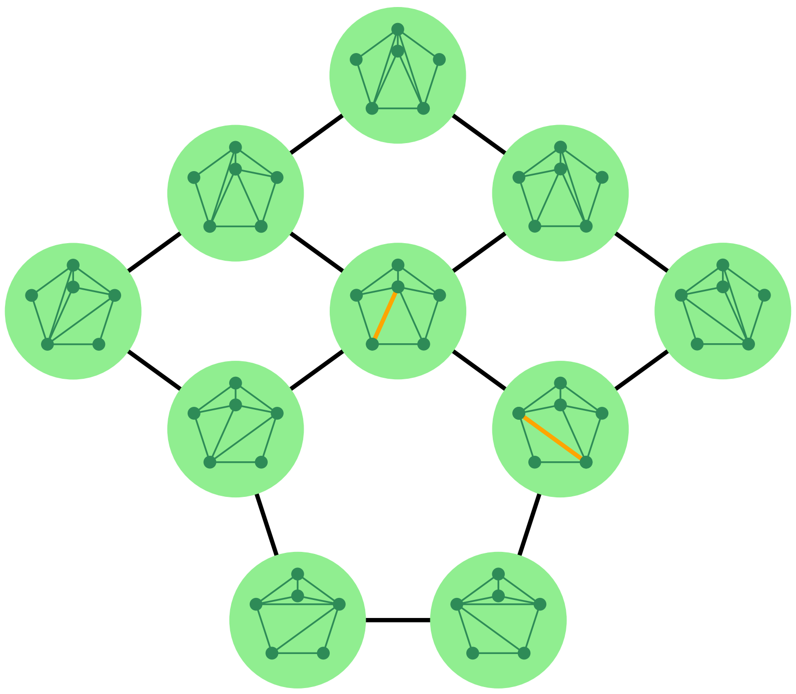


Figure 2.1: Example of a flip graph of point set with size six.

number of triangulations is exponential, the diameter of the flip graph is usually polynomial in the size of the point set.

Finding some flip sequence between two given triangulations. It is important to have algorithms that compute a (reasonably short) flip sequence between any two given triangulations. Often one can flip both triangulations into a canonical one. For this purpose, the Delaunay triangulation can be used for general point sets, the constrained Delaunay triangulation can be used for polygons, or a triangulation where all edges meet at a single vertex can be used for convex point sets.

Computing the flip distance and shortest paths in flip graphs. The *flip distance* between two triangulations is the minimum number of flips needed to transform one triangulation into the other, i.e., it is the length of the shortest path between the two triangulations in the flip graph. It is NP-complete to compute the flip distance for general point sets and polygons [101; 7], and the problem is open for convex point sets.

Computing the number of triangulations and generating random triangulations. Generating a triangulation of a point set uniformly at random by random flipping and a related question of counting the number of triangulations are open. Although it has been established that point set triangulations contain a linear number of flippable edges, more results on connectivity of flip graphs and mixing are needed. The existing bounds on the number of point set triangulations are also not tight.

Optimizing triangulation properties. Sometimes it is desirable to construct triangulations optimizing certain measures. There exist algorithms that use triangulation reconfiguration (either flipping or edge insertion that is introduced in Section 2.3) to construct triangulations minimising, maximising or approximating some of the following: total length of edges, maximum or minimum angle, triangle area, height, edge length, eccentricity and others. Finding efficient algorithms to optimize other triangulation properties is mostly open.

As already hinted at in Chapter 1, reconfiguring triangulations via flips has been important in various applications as well as related to the study of other mathematical objects. Triangulation flipping has been used in the study of combinatorial structures, such as associahedra and rotations in binary trees [129], in graph untangling, or in mixing of triangulation walks [107]. In geometric graph theory triangulations further give useful information on other kinds of planar graphs, such as non-crossing spanning trees, as these are subgraphs of triangulations. Flips are also important in practice, for example in computer graphics, to generate triangular meshes approximating shapes; and for finding triangulations that optimize certain quality measures [22; 54]. The survey by Bose and Hurtado [30] discusses these and many other aspects of flips.

Despite the long-term interest, some fundamental questions about triangulations remain hard to answer. The reason may partially lie in the fact that the number of triangulations is exponential in the size of the point set. Thus it quickly becomes impossible to construct flip graphs explicitly, and tools like polynomial time shortest paths graph algorithms must be replaced by other more sophisticated methods.

We give a short survey of the classic results on flip graphs in Section 2.3. To do that, let us start with a short review of basic triangulation properties in the next section.

2.2 Some triangulation properties

In this section we summarise basic triangulation properties. A friendly introduction can be found, for example, in a textbook by Devadoss and O'Rourke [50].

From the definition of a triangulation, it is easy to see that a triangulation of a polygon or a point set P always exists. Also, any set S of pairwise non-crossing edges with vertices from P can be extended into a triangulation – this is known as a *constrained triangulation* (with respect to S). A triangulation forms a maximal planar graph and it can be shown that it is a subdivision of the convex hull of the point set or of a polygon into triangular faces. [50].

The number of edges and of triangles is constant across all triangulations of a single point set or polygon. All triangulations of a point set P in general position with h points on the convex hull and k points inside contain $h-2+2k$ triangles and $h-3+3k$ internal edges. In case of an n-vertex polygon Q, triangulations always have $n-2$ triangles and $n-3$ internal edges. These counts do not include edges on the convex hull of P (or on the polygonal boundary of Q), since such edges belong to every triangulation of P (or Q) and are always non-flippable. Those relations can be deduced from the Euler's formula, stating that for any connected planar graph,

$$\#\,vertices - \#\,edges + \#\,faces = 2.$$

In particular, a maximal planar graph with a triangular outerface on n vertices has $2n-4$ faces (including the outerface) and $3n-6$ edges (including the edges on the convex hull).

Simple algorithms for triangulating include triangle splitting and an incremental algorithm for point sets, or an ear-trimming algorithm for simple polygons without holes [50].

An *ear* of a polygon is a triangle abc such that the points a, b, c lie consecutively on polygonal boundary and the diagonal ac lies in polygon's interior. It can be proven that any simple polygon without holes with more than three vertices contains at least two ears [50]. Then to triangulate a polygon, iteratively "cut off" an ear until a triangulation is obtained.

The triangle splitting algorithm first finds and triangulates the convex hull of a given point set as if it was a polygon. The remaining interior points of the point set are then, one at a time, connected to the three vertices of the triangle they are in, until a triangulation is obtained.

Finally the incremental triangulation algorithm scans the points of a point set in order of their x-coordinates. The first three points create a triangle and every next point is connected to all previous previous points that are visible from it, resulting in a triangulation.

All of the above triangulation algorithms have naive implementations in $O(n^2)$ time, where n is the number of vertices. It was proven that a simple polygon can be triangulated in $O(n)$ time [42], but this algorithm requires a very cumbersome implementation. In practice, to obtain a generic triangulation (i.e. a triangulation not optimised for any specific property), $O(n \log n)$ time algorithms are used [50].

The problem of deciding whether a set of edges contains a triangulation of the point set was shown to be NP-complete [99].

Regarding triangulation embeddings, by a theorem of Whitney (see, for example [112]) any 3-connected planar graph has a unique plane embedding. This means that for any 3-connected triangulation T, the set of faces in any plane embedding of its (abstract) graph is the same and is equal to the set of triangles in the original triangulation. Moreover, given the graph of T, the plane embedding can be reconstructed, and drawn with straight-line edges, by using a drawing algorithm for planar graphs, for example, the one devised by Tutte [136], or some of its improvements [126; 48], where any face can be made into an outer face of such a drawing. For an n-vertex planar graph, efficient versions of the graph drawing algorithms run in linear time [126; 48]. Finally, if a triangulation is 4-connected, then by Tutte's theorem it has a Hamiltonian circuit [135].

A triangulation that is special in several respects is a *Delaunay triangulation*; we denote it by DT. An introduction to its basic properties can be found, e.g., in the textbook by Devadoss and O'Rourke [50]. The Delaunay triangulation maximises the minimum angle present in any triangle of a triangulation over all possible triangulations of a given point set P. Assuming general position, an edge ab, with $a, b \in P$, belongs to $\mathrm{DT}(P)$ if and only if there exists an empty circle through a and b, i.e. a circle that passes through points a and b and contains no other points of P in its interior. The Delaunay triangulation can be built in time $O(n \log n)$. A remarkable fact is that if a triangulation of a point set is not Delaunay, then there always exists a local improvement step: a *Delaunay flip* (sometimes also called *Lawson flip*). An edge ac represents a Delaunay flip for an edge bd if ac crosses bd and $ac \in \mathrm{DT}(\{a, b, c, d\})$. A sequence of Delaunay flips cannot be cyclic and so, irrespective of which triangulation of P we start with, the flip sequence produces the Delaunay triangulation. This algorithm is also called a *Lawson flip algorithm.* For details, see the original results by Lawson in [97; 98] or a textbook explanation in, e.g., [50].

Analogous results exist for constrained triangulations. Given a set S of pairwise non-crossing edges on a point set P, the *constrained Delaunay triangulation of P and S* maximizes the minimum angle present in any triangle of a triangulation over all possible triangulations of P containing the edges in S [54]. It is defined analogously to the standard Delaunay triangulation, except that the empty circle property of an edge tolerates points to be inside the circle, as long as they are not visible from the defining edge. More precisely, points $a, b \in P$ are said to be *visible from each other* if the segment ab does not cross any of the constrained edges in S (assuming the general position of points). Then the edge ab belongs to the *constrained Delaunay triangulation of P and S* if either $ab \in S$ or if the points a and b are visible from each other and there is a circle through a and b such that each point inside this circle is invisible from every point of ab. Note that if the set of constrained edges is empty, the constrained Delaunay triangulation becomes the standard Delaunay triangulation. The constrained Delaunay triangulation can be built in time $O(n \log n)$, see [44; 22]. As before, if a triangulation T of P constrained to S is not constrained Delaunay, it is possible to define a local improvement step. Call an edge $ab \in T$ *locally constrained Delaunay* if either ab is a constrained edge from S, or a convex hull edge, or if the circle defined by points a, b, c does not contain the point d, where abc and abd are triangles of T. It can be shown that, irrespective of which constrained triangulation we start with, flipping the non-locally-constrained-Delaunay edges cannot be cyclic and so, after $O(n^2)$ flips produces a triangulation in which every edge is locally constrained Delaunay. Finally, one can prove that such a triangulation is the constrained Delaunay triangulation. For details, see, for example [54].

With the above we are ready to discuss the connectivity and other properties of flip graphs, which we do in the next section.

2.3 Review of flip graph properties and triangulation reconfiguration

In this section we review classical results on triangulation reconfiguration as introduced in Section 2.1. We survey the topics of connectivity, diameter, flip sequences, flip distance and other aspects of flipping for planar point sets, simple polygons, the special case of convex polygon, as well as mention the main results on combinatorial triangulations. An overview of the properties is given in Table 2.1. For further results on triangulation flipping in geometric, as well as combinatorial settings, see the survey by Bose and Hurtado [30]. Finally, apart from classic flipping we also briefly discuss the main results on other ways of reconfiguring triangulations – simultaneous flips and edge insertion.

Flip graphs of planar point sets

Lawson [97] proved the foundational result that any triangulation can be transformed into any other triangulation of the same point set via a sequence of flips. His second proof of this result [98] uses the approach that is more widely known—showing that any triangulation can be flipped to the Delaunay triangulation as described in Section 2.2, which then acts as a “hub” through which we can flip any triangulation to any other. Thus the flip graph of a planar point set is always connected. The diameter of the flip graph is known to be $\Theta(n^2)$ where n is the size of the point set. The upper bound was proved by Lawson [98] and the lower bound by Hurtado et al. [83]. Their paper also discusses a result stating that for point sets the diameter of the flip graph depends on the number of convex layers, l, of the point set and states that the diameter is $O(ln)$ [83].

Analogous results hold for flip graphs of constrained triangulations. Several times in the next sections we will use the result that if two triangulations of the same point set have a subset, S, of constrained edges in common, then there is a sequence of flips that transforms one triangulation into the other, without ever flipping any edge

$n =$ # of triangulation vertices	Flip graph connectivity proven by	To reconfigure triangulations, can use	Flip graph diameter d	Computing the flip distance	Number of triangulations
Planar point sets	Lawson [97]	Delaunay triangulation	$\Theta(n^2)$ Lawson [98]; Hurtado et al. [83]	NP-hard APX-hard Lubiw, Pathak [101]; Pilz [118]	$\Omega(8.65^n)$ $O(30^n)$ Dumitrescu et al. [51]; Sharir, Sheffer [128]
Constrained triangulations	Dyn et al. [52]	constrained Delaunay triangulation	$\Theta(n^2)$	NP-hard Aichholzer et al. [7]	
Simple polygons	Bern, Eppstein [22]	constrained Delaunay triangulation	$\Theta(n^2)$ Bern, Eppstein [22]; Hurtado et al. [83]	NP-hard Aichholzer et al. [7]	
Convex point set	Wagner [142]	triangulation with all edges meeting at a single vertex	$2n - 10$ for $n > 12$ Sleator et al. [129]; Pournin [121]		Catalan number $\frac{1}{n-1}\binom{2(n-2)}{n-2}$
Combinatorial triangulations	Wagner [142]	Wagner's canonical form	$\frac{7n}{3} - 34 \leq d \leq 5n - 23$ Frati [63]; Cardinal et al. [39]		

Table 2.1: Overview of flip graph properties for triangulations of size n.

of S, i.e., the edges in S remain fixed throughout the flip sequence. This was first proved by Dyn et al. [52], and can alternatively be proved using constrained Delaunay triangulations as explained in Section 2.2. The diameter of the constrained flip graph is again $\Theta(n^2)$. For details, see, for example [22; 54].

Given two point set triangulations, computing some flip sequence transforming one into the other is easy – just use Lawson's algorithm from above. The situation changes if one wants to find the shortest flip sequence between the triangulations. In general this is possible only for very small n, by explicitly constructing the flip graph and running a shortest path algorithm. Computing the distance in the flip graph between two given triangulations of a point set is NP-hard [101], and even APX-hard [118]. It has recently been shown to be fixed-parameter tractable [90].

If the point set contains no *empty convex pentagon* (i.e. no five points form a convex polygon empty of other points of the point set), Eppstein gave an $O(n^2)$-time algorithm for computing the flip distance. For arbitrary point sets, this algorithm computes a lower bound on the flip distance [57].

Yet another result on flip distances was given by Hanke et al. [71] who upper-bound the flip distance between two triangulations of a point set by the number of edge intersections when the two triangulations are drawn on top of each other.

Any triangulation of an n-vertex point set in general position was proven to have at least $\lceil (n-4)/2 \rceil$ flippable edges [83]. Hence, the vertex degrees in the corresponding flip graph must all be of order $\Theta(n)$, i.e. roughly uniform.

Also notice that the flip graph of a point set is usually not bipartite. This is because whenever the point set contains an empty convex pentagon, the five flips in Figure 2.3 define a 5-cycle in the flip graph.

The exact numbers of triangulations are unknown. For n points in general position, the number of triangulations varies, depending on the combinatorial type of the point set, and there has not been established a tight bound. In [128], it was proved that the number of different triangulations is $O(30^n)$. There exist point sets with $\Omega(8.65^n)$ triangulations [51]. Also, any n-vertex point set in general position has at least $\Omega(2.631^n)$ geometric triangulations [2]. As compared to a point set in convex position (see below), the number of triangulations of a general point set can be smaller or larger than the Catalan number C_{n-2} but it will always be exponential.

Flip graphs of simple polygons

Bern and Eppstein [22] showed that Lawson's proof [98] of flip graph connectivity in which each triangulation is transformed into a canonical form applies to simple polygons. One just needs to use the constrained Delaunay triangulation, where the set of constrained edges contains the polygonal boundary. Hence, it is always possible to flip between two triangulations of the same polygon.

By an analogous argument as for point sets, the diameter of the flip graph is $\Theta(n^2)$ [22; 83]. Hurtado et al. [83] express the diameter as a function of the number of its reflex vertices. In particular, for an n-vertex simple polygon Q with k reflex vertices, the diameter is $O(n + k^2)$ [83].

The problem of computing the flip distance remains NP-hard for triangulations of a simple polygon [7].

The number of flippable edges in a triangulation of a simple polygon with k reflex vertices is at least $n - 3 - 2k$, proved in [83].

A simple polygon on n vertices can have between 1 and the Catalan number C_{n-2} of triangulations, where the latter occurs when the vertices lie in a convex position (see [50] and below).

For not necessarily simple polygons, Eppstein [59] showed that *counting* the number of triangulations of a given polygon is #P-complete.

Flip graphs of the convex point set

Worth discussing is the special case of triangulations of convex point sets. In this case triangulations correspond to binary trees, and a flip corresponds to a rotation. A little survey on binary trees and rotations as well as the correspondence between triangulations and binary trees can be found in [129].

Unlike for the general point sets, the exact flip graph size is known: for a convex n-gon there exist C_{n-2} different triangulations where

$$C_{n-2} = \frac{1}{n-1}\left(\begin{array}{c} 2(n-2) \\ n-2 \end{array}\right) = \Theta(n^{-\frac{3}{2}} \cdot 4^n)$$

is the Catalan number, see [50]. The flip graph of an n-vertex point set in convex position is connected, Hamiltonian [103; 81] and, moreover, it is the 1-skeleton of an $(n-3)$-dimensional polytope called the *associahedron*, see [50].

The diameter of the flip graph is equal to $2n-10$ for all convex point sets of size $n > 12$, proved in [129; 121]. The upper bound was proven by Sleator et al. [129] directly by transforming any pair of triangulations into a triangulation in which all edges meet at a single vertex. For the lower bound, given a pair of triangulations of the same point set, Sleator et al. form a polyhedron by gluing the triangulations along the outer boundaries and show that being able to triangulate the polyhedron with k tetrahedra corresponds to a flip sequence of length k between the two triangulations. They use hyperbolic geometry to generate polyhedra that require many tetrahedra to be triangulated and prove the lower bound of $2n-10$ on the flip graph diameter as long as n is sufficiently large. About tweny-five years later Pournin [121] reproved the result by purely combinatorial methods, and, moreover, was able to prove the lower bound of $2n-10$ for all point sets of size $n > 12$. Note that in terms of the associahedron dimension, this result states that a d-dimensional associahedron has diameter $2d-4$ whenever $d > 9$.

Given two convex point set triangulations, computing some flip sequence reconfiguring one into the other is easy. The situation is again different if one requires the shortest flip sequence or the flip distance. The complexity status of computing the flip distance for triangulations of convex polygon is open.

Sleator et al. [129] give some results regarding the shortest flip sequence. In particular, they prove that if two given triangulations of the same convex point set have some edges in common, then a shortest flip sequence between them never flips these edges. Moreover, if such an edge was flipped, the resulting flip sequence would be by at least two flips longer than the shortest flip sequence. Another result states that if flipping an edge e in the first triangulation increases the number of edges that the two triangulations have in common then there exists a shortest flip sequence in which the first flip flips the edge e. All of these results are proven by so-called *triangulation normalization* [129].

Flip graphs of combinatorial triangulations

A *combinatorial triangulation* is a simple maximal planar graph in which a clockwise order of incident edges is specified around every vertex. There is apriori no specified embedding. No multiple edges or loops are allowed. If embedded as a triangulation, the faces, including the outerface, would all be 3-cycles. There would be exactly $3n-6$ edges and $2n-4$ faces, where n is the number of vertices. Since all maximal planar graphs are 3-vertex-connected, by a result by Whitney (see Section 2.2) the embedding of a combinatorial triangulation in plane is unique.

Flipping is well-defined in a combinatorial triangulation, since removing an edge ac determines a 4-cycle $abcd$, where abc and acd are 3-cycles of the triangulation. Then ac can flip to bd as long as the edge bd is not yet present in the triangulation. A flip can result in an isomorphic triangulation and since no loops are allowed, such a flip would not correspond to an edge in the flip graph.

The fact that the flip graph of combinatorial triangulations is always connected was first established by Wagner by transforming the triangulations to the so-called Wagner canonical triangulation form [142].

The diameter of the flip graph was proved to be of order $\Theta(n)$ where n is the number of vertices in the triangulation; Sleator et al. [130] proved the upper bound, while the lower bound follows from reconfiguring certain classes of triangulations to Wagner's canonical form. The best known upper bound on the diameter is currently $5n-23$ by Cardinal et al. [39] and the best lower bound $\frac{7n}{3}-34$ by Frati [63].

Sleator et al. [130] also discussed combinatorial triangulations with vertex labels. These can be seen as a triangulation form in between the classic combinatorial triangulations and the geometric planar triangulations. In this case they proved the diameter of the flip graph for an n-vertex triangulation to be $\Theta(n \log n)$.

For a survey on combinatorial triangulations, see [32].

Optimizing triangulation properties, silmultaneous flips and edge insertion

So far we discussed properties of flip graphs and flipping. Flips provide a one-step reconfiguration, transforming a triangulation into an adjacent one in the flip graph. In this section we review reconfiguration methods that enable one to make bigger 'jumps' in the flip graph, between triangulations that are not necessarily adjacent. These are 'simultaneous flips' and 'edge insertion'.

One motivation for studying these types of reconfigurations is that making bigger steps in the flip graph is useful in optimizing some quality measures in triangulations. In the case of optimizing a measure by classic flipping, the flip sequences can get stuck in triangulations that are only locally optimal, and miss a triangulation that optimizes the given measure over all triangulations.

Simultaneous flipping

Simultaneous flipping was introduced by Hurtado et el. [82]. In the geometric setting a *simultaneous flip* can flip a set of edges in parallel, as long as each edge is flippable individually and no two of these edges are incident to the same triangle.

Galtier et al. [64] proved that any triangulation of an n-vertex point set or a simple polygon can be reconfigured to any other triangulation in $\Theta(n)$ simultaneous flips, while for triangulations of a convex point set this quantity changes to $\Theta(\log n)$ flips.

In the combinatorial setting, in addition to requiring that no two of the edges to be flipped are incident to the same triangle, we also require that the simultaneous flip does not create any multiple edges. Note that it is possible that the simultaneous flip includes an edge that could not be flipped individually. Bose et al. [29] proved that to reconfigure a combinatorial triangulation into any other, $\Theta(\log n)$ simultaneous flips are sufficient and sometimes necessary.

A comparison of bounds on flip sequence lengths between classical and simultaneous flipping for the different triangulation types can be found in the first two columns of Table 2.2 on page 57. In all cases simultaneous flipping saves a non-constant number of flips.

Edge insertion

As was already discussed in previous sections, important (triangulation) reconfiguration problems involve computing the flip graph diameter and short flip sequences. Another aspect of triangulations that applications such as mesh generation often ask for is to generate a triangulation optimizing certain quality measures. The main goal is the following: given a quality measure, identify a triangulation that optimizes the measure over all possible triangulations of a given point set.

There are two caveats one needs to consider. Firstly, what constitutes a ‘good’ quality measure is not clearly defined and depends on an application. A general rule in computer graphics and elsewhere is to require that the triangulations do not contain ‘long and thin’ triangles [21; 50]. This can be ensured by minimizing, for example, the maximum angle, maximum edge length, maximum triangle eccentricity, maximum area, maximum radius of a circumcircle or a smallest enclosing circle of a triangle in the triangulation [21; 55; 138; 43]. Usually, the property to be optimised depends on a single triangle and the quality of a triangulation is considered to be the quality of its worst triangle [21]. Secondly, different measures are typically optimised by different triangulations. For example, the Delaunay triangulation introduced in Section 2.2 maximizes the minimum triangle angle present in a triangulation, over all triangulations of the point set. It also minimizes the maximum circumradius or the maximum radius of the smallest enclosing circles of triangles in a triangulation [43], however, if one wishes to, for example, minimize the maximum angle, then the Delaunay triangulation may not be the globally optimal triangulation [56].

A reasonable approach to obtain an optimal triangulation is to start with an arbitrary triangulation of the point set and, through a sequence of improvement steps, reach a triangulation globally optimizing the given criterion. One has to guarantee that whenever the current triangulation is not yet optimal, an improvement step is possible. At the same time the algorithm should be able to find the global optimum in polynomial time. Lawson’s edge-flipping algorithm introduced in Section 2.2 reaches in this way the Delaunay triangulation. For other measures, however, Bern et al. [21] show that edge flipping can get stuck at a locally optimal triangulation.

Edelsbrunner et al. [56] introduce an alternative to edge-flipping: *edge insertion*.

Given a triangulation T of a point set P and two points $u, v \in P$, the edge uv is *inserted into* T by adding uv into T, deleting all edges of T that cross uv and retriangulating the created polygonal regions in a suitable way. Similarly to edge flipping, edge insertion provides a way of generating a new, locally changed triangulation. Unlike edge flipping, it makes 'bigger jumps' between triangulations in a flip graph, and so it might also be better able to leave local maxima.

Edge insertion is associated with decreasing angle sizes at the endpoints of the inserted edge. Triangular properties that are at least partially related to angle sizes are, for example, the maximum angle in a triangulation, minimum height, triangle eccentricity, and possibly others. Bern et al. [21] formulate a general *edge insertion paradigm.* This gives a polynomial-time edge insertion algorithm that, if applied to a triangle measure μ satisfying certain specified criteria, is guaranteed to output a triangulation globally optimising μ over all triangulations of a given polygon or a point set. In particular, the authors [56; 21] prove that the maximum angle and the negative of the minimum triangle height in a triangulation satisfy the conditions of the paradigm and, hence, can be optimized via edge insertion.

For a survey on optimizing triangulation properties, including the edge insertion method, see [22].

2.4 Introduction to edge-labelled triangulations

For the remainder of this chapter we turn our attention to one of the main objects of study within this thesis: the edge-labelled triangulations of point sets. After introducing the necessary concepts and background, we present a proof of the Orbit Conjecture that characterizes when two edge-labelled triangulations are connected by a sequence of flips.

More precisely, in the labelled setting each edge of a triangulation has a label, and a flip transfers the label of the removed edge to the new edge. It turns out that it is no longer true that every labelled triangulation of a point set can be reconfigured to every other labelled triangulation via a sequence of flips, but we characterize when this is possible. There is an obvious necessary condition: for each label l, if edge e

has label l in the first triangulation and edge f has label l in the second triangulation, then there must be some sequence of flips that moves label l from e to f, ignoring all other labels. Bose, Lubiw, Pathak and Verdonschot formulated the “Orbit Conjecture”, which states that this necessary condition is also sufficient, i.e. that *all* labels can be simultaneously mapped to their destination if and only if *each* label individually can be mapped to its destination. We prove this conjecture. Furthermore, we give a polynomial-time algorithm to find a sequence of flips to reconfigure one labelled triangulation to another, if such a sequence exists, and we prove an upper bound of $O(n^7)$ on the length of the flip sequence.

Our proof uses the topological result that the sets of pairwise non-crossing edges on a planar point set form a simplicial complex that is homeomorphic to a high-dimensional ball (this follows from a result of Orden and Santos; we give a different proof based on a shelling argument). The dual cell complex of this simplicial ball, called the *flip complex*, has the usual flip graph as its 1-skeleton. We use properties of the 2-skeleton of the flip complex to prove the Orbit Conjecture.

Although there is a rich literature on associahedra and on cell complexes associated with triangulations of point sets, there are very few other combinatorial results that require topological proofs, as our proof of the Orbit Theorem seems to. One other example is the study of graph colourings and their reconfiguration where topological properties of the so-called box complex has been used. See Wrochna [146] for a list of works that used the box complex to obtain lower bounds on chromatic numbers of various families of graphs, as well as for some new ways of using the box complex for colouring reconfiguration.

We start in Section 2.5 with a general motivation and basic definitions related to edge-labelled triangulations. Section 2.6 states the Orbit Theorem and gives a brief summary of previous relevant results. In Sections 2.7 - 2.10 we prove the Orbit Theorem. The proof is subsequently extended in Section 2.11 to cover constrained edge-labelled triangulations.

2.5 Basic definitions and motivation for edge-labelling

Despite the extensive work on flips in triangulations, it is only recently that the question of where edges go under flip operations has been investigated. This is formalized by attaching a label to each edge in a triangulation. A *labelled triangulation* $\mathcal{T}$ of a planar point set P is a pair (T, ℓ) where T is a triangulation of P and ℓ is a *labelling function* that maps the edges of T one-to-one onto the labels $1, 2, \ldots, t_P$. Here t_P is the number of (interior) edges in any triangulation of P. Two labelled triangulations (T_1, ℓ_1) and (T_2, ℓ_2) are the same if both the unlabelled triangulations and the corresponding labels coincide, i.e. $T_1 = T_2$ and $\ell_1 = \ell_2$. When we perform a flip operation on $\mathcal{T}$, the label of the removed edge is transferred to the new edge. The *labelled flip graph* has a vertex for every labelled triangulation of the point set and an edge when two labelled triangulations differ by a flip.

Edge-labelled triangulations were introduced independently in several papers, by Araujo-Pardo et al. [15], Bose et al. [31] and Espinas et al. [60]. The idea is that reconfiguring one triangulation into another moves the edges and labelling them enables us to track where the individual edges go during this process. This may reveal more about the structure and flip sequences in the (unlabelled) flip graph. For example, knowing the start and end position of each edge was used by Eppstein [57] in developing an algorithm to compute flip distances between triangulations of point sets that contain no *empty convex pentagon*, i.e. point sets where no 5 points form a convex polygon empty of other points in its interior.

Espinas et al. [60] used edge labels in an algorithm that reduces the length of an existing flip sequence σ between (unlabelled) triangulations of point sets (or between combinatorial triangulations). The reduced flip sequence reconfigures the initial triangulation into the same triangulation as the original flip sequence σ, up to a permutation of labels. The authors show that if T_1 and T_2 are triangulations of the convex point set, and the flip sequence σ reconfigures T_1 into T_2 , then σ can be transformed into any other flip sequence, and in particular into the shortest flip sequence between triangulations T_1 and T_2 by applying some number of their reduction operations [60]. More results on the convex point set appeared in [15; 31] and are discussed in the next section.

Another potential application of edge labelling is to reconfiguration problems of planar graphs as these are subgraphs of triangulations or, as a means to carry additional information about the triangulation [60].

In what follows we work with edge-labelled triangulations as defined above; but we remind the reader that there exist other types of triangulation labellings, as discussed in Section 1.3.1.

Throughout, we fix a set P of n points in general position. We say that edges e and f lie in the same *orbit* if we can attach label l to e in some triangulation and apply some sequence of flips to arrive at a triangulation in which edge f has label l. The orbits are exactly the connected components of a graph that Eppstein [57] called the *quadrilateral graph*—this graph has a vertex for every one of the possible $\binom{n}{2}$ edges formed by point set P, with e and f being adjacent if they cross and their four endpoints form a convex quadrilateral that is empty of other points (i.e. whenever e and f can be flipped into each other in some triangulation of P). In particular, this implies that there is a polynomial-time algorithm to find the orbits.

The orbits can be very different depending on P. For a point set in convex position, all the non-convex hull edges are in a single orbit [31]. At the other extreme there are point sets like grids that contain no empty convex pentagon. Eppstein [57] proved that a point set P with no empty convex pentagon has the property that in any triangulation, the edges are all in distinct orbits. Then, if one (unlabelled) triangulation of P needs to be reconfigured into another, every edge has a uniquely determined target edge in the final triangulation onto which it must eventually flip. If an edge labelling of the triangulations does not respect these orbits, then no flip sequence exists between the edge-labelled triangulations.

This illustrates a further difference between labelled and unlabelled triangulations: unlike in the unlabelled case, a flip graph of labelled triangulations can be disconnected. It happens whenever the point set or polygon has multiple orbits: for example, in point sets with no empty convex pentagon, but also whenever some edge is fixed or can be flipped only within some restricted region, see for example Figure 2.2.

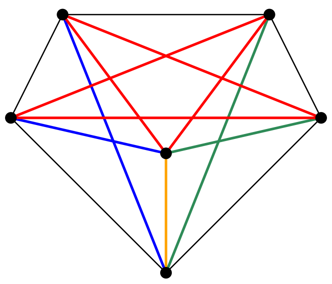


Figure 2.2: Point set with a disconnected flip graph of labelled triangulations. Colours of individual diagonals indicate the orbit that they belong to. For example, no blue diagonal will ever be able to flip to replace any of the red diagonals.

2.6 Orbit Conjecture and related results

Orbits tell us where each individual edge label can go, but not how they combine. One of the main questions that we address in this thesis is:

When is there a sequence of flips to reconfigure one labelled triangulation of point set P to another labelled triangulation of P?

A necessary condition is that, for each label l, the edges with label l in the two triangulations must lie in the same orbit. Bose et al. [31] conjectured that this condition is also sufficient. As our main result on edge-labelled triangulations we prove this "Orbit Conjecture" and strengthen it by providing a polynomial-time algorithm and a bound on the length of the flip sequence.

Theorem 1 (Orbit Theorem)**.** *Given two edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ of a point set, there is a flip sequence that transforms one into the other if and only if for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_2$ having label l belong to the same orbit. Furthermore, there is a polynomial-time algorithm (with $O(n^8)$ being a crude bound on its run-time) that tests whether the condition is satisfied, and if it is, computes a flip sequence of length $O(n^7)$ to transform $\mathcal{T}_1$ to $\mathcal{T}_2$.*

The Orbit Theorem holds for combinatorial triangulations [31], and for pseudotriangulations [33]. In both these cases there is a single orbit, the labelled flip graph is connected and so any reconfiguration of labels can be realized by flips. There are also some related results using variants of the flip operation, for example, Cano et al. [38] reconfigured edge-labelled non-maximal plane graphs by "rotating" edges around one of their endpoints; again there is a single orbit. A related result where there are multiple orbits is an analogue of the Orbit Theorem for edge-labelled spanning trees of an underlying graph by Hernando et al. [75]. A flip in this case is an exchange of a spanning tree edge, carrying over the respective label. This was generalized to an Orbit Theorem result for labelled (or “ordered”) bases of a matroid—one labelled basis can be turned into another labelled basis via basis exchange steps if and only if elements with the same label lie in the same connected component of the matroid [102].

In the geometric triangulations setting, the conjecture had been known to hold for point sets with no empty convex pentagon based on the work by Eppstein [57]: combining the result that each triangulation of such a point set contains one edge per each orbit with the fact that unlabelled triangulations can always be reconfigured by a sequence of flips gives the Orbit Theorem for this class of point sets. Bose et al. [31] formulated the Orbit Conjecture and proved it for the special cases of triangulations of any convex or spiral polygon and in each case found tight bounds on the diameter of connected components in the corresponding labelled flip graph. Bose et al. [31] also found the best known lower bound on the diameter of a connected component of the labelled flip graph for a point set, namely $\Omega(n^3)$. There is a large gap between this lower bound and our upper bound of $O(n^7)$ stated in the Orbit Theorem. Finally, after publishing our proof of the Orbit Theorem, Pilz [119] was able to modify hardness proofs from the unlabelled setting and showed that computing the flip distance between edge-labelled triangulations of point sets is APX-hard and between edge-labelled triangulations of simple polygons it is NP-hard.

We now describe some of the above results in more detail.

Orbit Theorem for convex point sets. As we mentioned earlier, all edges spanned by a convex point set (ignoring the convex hull edges that can never flip) lie in a single orbit, and so proving the Orbit Conjecture amounts to proving that the labelled flip graph is connected. Bose et al. [31] showed that this is indeed the case and gave tight bound of $\Theta(n \log n)$ on the diameter of the flip graph (where the lower bound $\Omega(n \log n)$ comes from an argument by Sleator et al. [130]). The authors also proved that there exists a polynomial time approximation algorithm with factor $O(\log n)$ for computing the flip distance for edge-labelled triangulations of convex point sets [31]. In the case when simultaneous labelled flips are allowed, Bose et al. [31] prove that $O(\log^2 n)$ simultaneous flips are always sufficient and $\Omega(\log n)$ simultaneous flips are sometimes necessary to reconfigure one labelled triangulation into another.

Araujo-Pardo et al. [15] independently proved the Orbit Conjecture for convex point sets, and introduced "colourful associahedra" which generalize associahedra to the setting of labelled (or coloured) triangulations. More precisely, as the flip graph of unlabelled triangulations of a convex point set is the 1-skeleton of the polytope called associahedron, Araujo-Pardo et al. showed that the flip graph of labelled triangulations is the 1-skeleton of a polytope that they call the colourful associahedron.

Orbit Theorem for spiral polygons. Bose et al. [31] fully characterized the orbits and proved the Orbit Conjecture for spiral polygons. These are simple polygons with a single reflex chain. Depending on the vertex set, they can have multiple orbits. The orbits were shown to consist of all the diagonals in the respective maximal *locally convex* subpolygons, where a subpolygon is locally convex if every four consecutive points on the convex part of its polygonal boundary form a quadrilateral empty of other points. The labelled flip graph was proved to be connected if the polygon is locally convex, and to contain multiple connected components otherwise. The authors proved a tight bound $\Theta(n^2)$ on the diameter of each of connected components of the flip graph, where n is the number of vertices in the polygon. Moreover, local convexity and, hence, the existence of a reconfiguring flip sequence can be checked in $O(n)$ time [31].

Lower bound on flip graph diameter for planar point sets and polygons. Bose et al. [31] constructed an example of a polygon on $2n + 2$ vertices, the so-called, augmented channel, whose labelled flip graph has diameter $\Theta(n^3)$. This is the best known lower bound on the diameter of a connected component of the labelled flip graph of a polygon or point set. It is larger than the corresponding bound for flip graphs of unlabelled triangulations.

A comparison of flip sequence lengths required to reconfigure triangulations in different settings as we discussed in Sections 2.3 – 2.6 is given in Table 2.2.

n = # of triangulation vertices	Unlabelled (classic) flips	Unlabelled simultaneous flips	Edge-labelled (classic) flips	Edge-labelled simultaneous flips
Planar point sets	$\Theta(n^2)$ Lawson [98]; Hurtado et al. [83]	$\Theta(n)$ Galtier et al. [64]	$O(n^7)$ (*) ($^\triangle$) $\Omega(n^3)$ (*) ($^\triangle$) Lubiw et al. [100]; Bose et al. [31]	$O(n^7)$ (*) $\Omega(n^2)$ (*) follows from ($^\triangle$) results
Simple polygons	$\Theta(n^2)$ Bern, Eppstein [22]; Hurtado et al. [83]	$\Theta(n)$ Galtier et al. [64]	$O(n^7)$ (*) $\Omega(n^3)$ (*) Lubiw et al. [100] and Section 2.11 in this thesis; Bose et al. [31]	
Spiral polygon	$\Theta(n)$ Hanke [70]		$\Theta(n^2)$ (*) Bose et al. [31]	
Convex point set	$2n-10$ for $n>12$ Sleator et al. [129]; Pournin [121]	$\Theta(\log n)$ Galtier et al. [64]	$\Theta(n\log n)$ Bose et al. [31]; Sleator et al. [130]	$O(\log^2 n)$ $\Omega(\log n)$ Bose et al. [31]
Combinatorial triangulations	$\frac{7n}{3}-34 \le d \le 5n-23$ Frati [63]; Cardinal et al. [39]	$\Theta(\log n)$ Bose et al. [29]	$\Theta(n\log n)$ Bose et al. [31]; Sleator et al. [130]	$O(\log^2 n)$ $\Omega(\log n)$ Bose et al. [31]; Bose et al. [29]

Table 2.2: Comparison of bounds on flip sequence lengths d required to reconfigure triangulations of different types by unlabelled/labelled and classical/simultaneous flips. In cases marked with (*) the flip graph may not be connected; these are the bounds on a connected component.

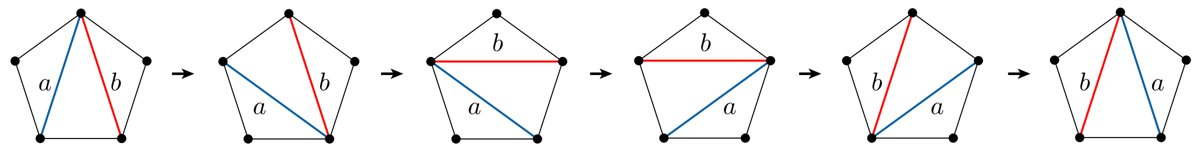


Figure 2.3: Five flips swap the edge labels (a and b) of two diagonals of a convex pentagon. In the flip graph these five flips form a 5-cycle.

The above proofs of the Orbit Theorem for the special cases (of convex point set, spiral polygon as well as of the combinatorial triangulation) show a general pattern: given a pair of labelled triangulations that should be reconfigured into each other, one can first ignore the labels and transform both triangulations into a specific canonical triangulation. Subsequently the labels are permuted, often by imitating some sorting algorihm. For example, relabelling edges in a convex polygon imitates quicksort while in spiral polygon it imitates the insertion sort.

Another insight to be gained from previous work is that empty convex pentagons in the point set seem to be crucial for swapping edge labels. Certainly, an empty convex pentagon provides a label swap—Figure 2.3 shows how the edge labels of two diagonals of an empty convex pentagon can be swapped by a sequence of five flips. In the other direction, the special cases of the Orbit Theorem that were proved by Bose et al. [31] for convex and spiral polygons involved moving pairs of labels into empty convex pentagons and swapping them there. Also, Eppstein [57] showed that in a triangulation of a point set with no empty convex pentagons, no permutations of edge labels are possible via flips.

Both of these insights turn out to be relevant and are used in the proof of the Orbit Theorem for general point sets that we present in the following sections.

2.7 A Proof of the Orbit Theorem: general outline

We present a proof of the Orbit Theorem 1 in Sections 2.7 – 2.10; and a summary is sketched in Figure 2.7. In this section we start by outlining the general plan for the proof.

The Orbit Theorem is stated for labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ that may have different edge sets, but—since we know how to use flips to change the edge set—

the crux of the matter is the special case where the two triangulations have the same edge set T but different label functions ℓ_1 and ℓ_2. In other words, we are given a permutation of the edge labels of a triangulation, and we seek a flip sequence to realize the permutation. Furthermore, since every permutation is a composition of transpositions, we concentrate first on finding a flip sequence to transpose (or "swap") two labels. This idea of reducing the problem to the case of swaps appears in [31].

The foundation of our proof is to make the intuition about empty convex pentagons rigorous. In particular, we show that the only elementary operation that is needed for label permutation is to transpose two labels by moving them into an empty convex pentagon and swapping them there. More formally, given a labelled triangulation $\mathcal{T} = (\mathcal{T}, \ell)$, an *elementary swap* of edges e and f in T is a transposition of the labels of e and f that is accomplished as follows: perform a sequence, σ, of flips on $\mathcal{T}$ to get to a triangulation $\mathcal{T}'$ in which the labels $\ell(e)$ and $\ell(f)$ are attached to the two diagonals of an empty convex pentagon; then perform the 5-flip sequence, π, that transposes these two labels; then perform the sequence σ^{-1}. We say that the sequence $\sigma\pi\sigma^{-1}$ *realizes* the elementary swap. Observe that the effect of $\sigma\pi\sigma^{-1}$ on $\mathcal{T}$ is to transpose the labels of e and f while leaving all other labels unchanged. We will prove that an elementary swap can always be realized by a flip sequence of length $O(n^6)$, and furthermore, that such a sequence can be found in polynomial time.

One of our main results is the following, from which the Orbit Theorem can readily be derived:

Theorem 2 (Swap Theorem). *In a labelled triangulation $\mathcal{T}$, two edges are in the same orbit if and only if there is an elementary swap between them.*

In order to prove Theorem 2, we use the following key result:

Theorem 3 (Elementary Swap Theorem). *Given a labelled triangulation $\mathcal{T}$, any permutation of the labels that can be realized by a sequence of flips can be realized by a sequence of elementary swaps.*

This theorem is proved using topological properties of the *flip complex*, whose 1-skeleton is the flip graph. A result of Orden and Santos [114] can be used to show that the flip complex has the topology of a high-dimensional ball (technically speaking, the flip complex is homotopy equivalent to a ball.). We give an alternate proof of this. We then use the 2-skeleton of the flip complex, and show that its 2-cells correspond to cycles in the flip graph of two types: quadrilaterals, which do not permute labels; and pentagons, which correspond precisely to the 5-cycles of flips shown in Figure 2.3. Then we prove the Elementary Swap Theorem by translating it into a result about decomposing closed walks in the flip graph into simpler *elementary walks*.

We now briefly describe the rest of our method after the Elementary Swap Theorem is established. In order to prove Theorem 2, we need one more ingredient about the structure of elementary swaps: we will show that any sequence of elementary swaps that moves the label of edge e to edge f can be "completed" to get the label of f back to e, and that, in fact, the resulting sequence provides an elementary swap of e and f.

The high-level idea of our proof of Theorem 2 is then as follows: From our hypothesis that two edges e and f lie in the same orbit, we show that there is a sequence of flips that permutes the labels of triangulation $\mathcal{T}$, taking the label of e to f. The Elementary Swap Theorem then gives us a sequence of elementary swaps to do the same (this is the significant step of the proof). Finally, from the structure of elementary swaps we can then find an elementary swap of e and f.

The proof of the Orbit Theorem 1 is organized in the next sections as follows. In Section 2.9 we prove the Elementary Swap Theorem using topological methods. In Section 2.10 we prove the properties of elementary swaps that were mentioned above. In top-down fashion, we begin in Section 2.8 by expanding on the high-level ideas, and proving the Orbit Theorem assuming the results in the later Sections 2.9 - 2.10.

Preliminaries and Definitions. We reiterate important assumptions and formal definitions that will be used in the proof of the Orbit Theorem: throughout, we assume a set of n point in general position in the plane. A point set determines $\binom{n}{2}$ *edges* which are the line segments between pairs of points. Two edges *cross* if they intersect in a point that is interior to at least one of the two edges. A *diagonal* of a convex polygon is an edge joining two points that are not consecutive on the polygon boundary.

Several times in our proofs we will use the result that if two unlabelled triangulations of the same point set have a subset, S, of *constrained* edges in common, then there is a sequence of flips that transforms one triangulation into the other, without ever flipping any edge of S, i.e. the edges in S remain fixed throughout the flip sequence. See Sections 2.2 - 2.3 for background on (constrained) triangulations.

2.8 Proof of the Orbit Theorem

In this section we prove the Orbit Theorem assuming the Elementary Swap Theorem (Theorem 3, proved in Section 2.9), and assuming the following two results on elementary swaps. The first result shows that every elementary swap can be realized by a relatively short flip sequence that can be found efficiently, and the second result gives us a way to combine elementary swaps so that, after moving e's label to f, we can get f's label back to e. These lemmas will be proved in Section 2.10.

Lemma 4. *If there is an elementary swap between two edges in a triangulation $\mathcal{T}$ then there is a flip sequence of length $O(n^6)$ to realize the elementary swap, and, furthermore, this sequence can be found in polynomial time.*

Lemma 5. *Let $\mathcal{T}$ be a labelled triangulation containing two edges e and f. If there is a sequence of elementary swaps on $\mathcal{T}$ that takes the label of edge e to edge f, then there is an elementary swap of e and f in $\mathcal{T}$.*

As we show in Section 2.10, a simple group-theoretic argument suffices to prove a weaker version of Lemma 5, namely, that under the stated assumptions, there is a sequence of elementary swaps exchanging the labels of e and f in $\mathcal{T}$. Proving the stronger version, which we need for our bounds on the length of flip sequences, requires using the properties of elementary swaps.

We prove the Orbit Theorem in stages, first Theorem 2 (the Swap Theorem that handles the case of swapping two labels in a triangulation), then the more general case of permuting edge labels in a triangulation, and finally the full result.

Proof of Theorem 2. The "if" direction is clear, so we address the "only if" direction. Suppose that $\mathcal{T} = (T, \ell)$ is the given edge-labelled triangulation and that e and f are edges of T that are in the same orbit. Then there is a sequence of flips that changes $\mathcal{T}$ to an edge-labelled triangulation $\mathcal{T}' = (T', \ell')$ where T' contains f and $\ell'(f) = \ell(e)$ (such a flip sequence can be found by, for example, following a path from e to f in the quadrilateral graph). We now apply the result that any constrained triangulation of a point set can be flipped to any other. Fix edge f and flip T' to T. Applying the same flip sequence to the labelled triangulation $\mathcal{T}'$ yields an edge-labelling of triangulation T in which edge f has the label $\ell(e)$. Thus we have a sequence of flips that permutes the labels of $\mathcal{T}$ and moves the label of e to f.

By the Elementary Swap Theorem (Theorem 3) there is a sequence of elementary swaps whose effect is to move the label of edge e to edge f (and possibly permute other labels). By Lemma 5 there is an elementary swap of e and f in $\mathcal{T}$. □

Theorem 6 (Edge Label Permutation Theorem)**.** *Let T be a triangulation of a point set with two edge-labellings ℓ_1 and ℓ_2 such that for each label l, the edge with label l in ℓ_1 and the edge with label l in ℓ_2 are in the same orbit. Then there is a sequence of $O(n)$ elementary swaps to transform the first labelling to the second. Such a sequence can be realized via a sequence of $O(n^7)$ flips, which can be found in polynomial time.*

Proof. The idea is to effect the permutation as a sequence of swaps. If every edge has the same label in ℓ_1 and ℓ_2 we are done. So consider a label l that is attached to a different edge in ℓ_1 and in ℓ_2. Suppose $\ell_1(e) = l$ and $\ell_2(f) = l$, with $e \neq f$. By hypothesis, e and f are in the same orbit. By Theorem 2 there is an elementary swap of e and f in (T, ℓ_1) which results in a new labelling ℓ_1' that matches ℓ_2 in one more edge (namely the edge f) and still has the property that for every label l, the edge with label l in ℓ_1' and the edge with label l in ℓ_2 are in the same orbit. Thus we can continue this process until all edge labels match those of ℓ_2. In total we use $O(n)$ elementary swaps. These can be realized via a sequence of $O(n^7)$ flips by Lemma 4. Furthermore, the sequence can be found in polynomial time. □

We can now prove the Orbit Theorem.

Proof of Theorem 1 (Orbit Theorem). The necessity of the condition is clear, and we can test it in polynomial time by finding all the orbits, so we address sufficiency. The idea is to reconfigure $\mathcal{T}_1$ to have the same underlying unlabelled triangulation as $\mathcal{T}_2$ and then apply the previous theorem. The details are as follows. Let $\mathcal{T}_1 = (T_1, \ell_1)$ and $\mathcal{T}_2 = (T_2, \ell_2)$. There is a sequence σ of $O(n^2)$ flips to reconfigure the unlabelled triangulation T_1 to T_2, and σ can be found in polynomial time. Applying σ to the labelled triangulation $\mathcal{T}_1$ yields a labelled triangulation $\mathcal{T}_3 = (T_2, \ell_3)$. Note that for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_3$ having label l belong to the same orbit. This is because flips preserve orbits (by definition of orbits). Thus by Theorem 6 there is a flip sequence τ that reconfigures $\mathcal{T}_3$ to $\mathcal{T}_2$, and this flip sequence can be found in polynomial time and has length $O(n^7)$. The concatenation of the two flip sequences, $\sigma\tau$, reconfigures $\mathcal{T}_1$ to $\mathcal{T}_2$, has length $O(n^7)$, and can be found in polynomial time. $\square$

2.9 Proof of the Elementary Swap Theorem

As mentioned in the introduction, we prove the Elementary Swap Theorem using topological properties of the *flip complex*, whose 1-skeleton (i.e. vertices and edges) is the flip graph. We will show that 2-cells of the flip complex correspond to 4- and 5-cycles in the flip graph.

The basic idea is as follows. We will translate the Elementary Swap Theorem to a statement about walks in the flip graph. The hypothesis of the Elementary Swap Theorem is that we have a sequence of flips that permutes the edge labels of a triangulation T. In the flip graph, this sequence corresponds to a closed walk w that starts and ends at triangulation T. Our main topological result is that the flip complex has a trivial fundamental group, which will imply that such a closed walk w can be decomposed into simpler *elementary walks*. Each elementary walk starts at T, traces a path in the flip graph, then traverses the edges of a 2-cell, then retraces the path back to T. The edge-label permutation induced by an elementary walk depends on the 2-cell. If the 2-cell is a 4-cycle, the permutation is the identity; and if the 2-cell is a 5-cycle, then the permutation is a transposition, and the elementary walk corresponds to an elementary swap. Altogether, this implies that the permutation induced by the closed walk w can be expressed as a composition of elementary swaps, which proves the Elementary Swap Theorem.

Before stating our main topological theorem, we first define the special cycles that will be shown to correspond to 2-cells of the flip complex. In the same way that an edge of the flip complex corresponds to two triangulations that differ on one edge, every 2-cell of the flip complex corresponds to a set of triangulations that differ on two edges. Define an *elementary 4-cycle* to be a cycle of the flip graph obtained in the

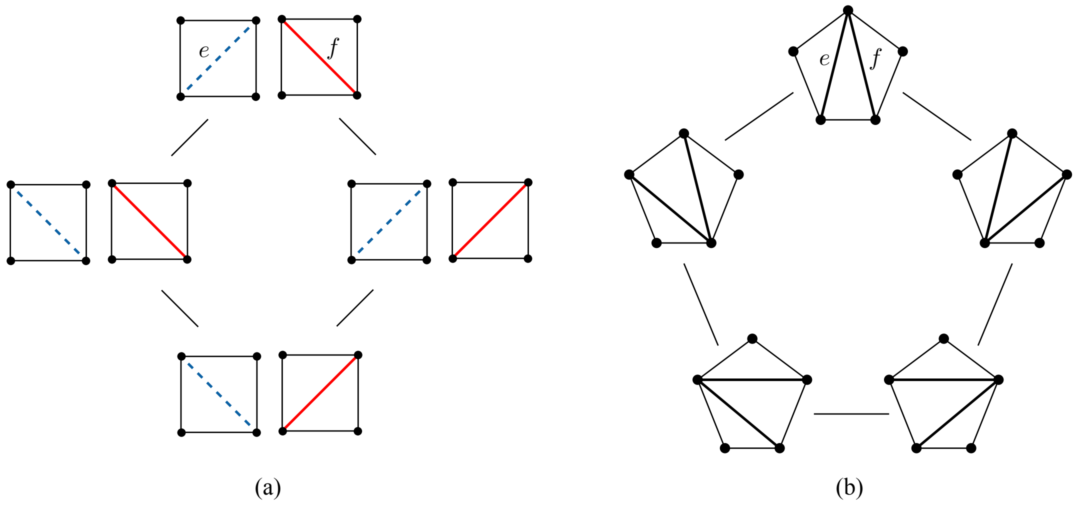


Figure 2.4: (a) Triangulations that differ in the diagonals of two internally disjoint quadrilaterals form an *elementary* 4*-cycle* in the flip graph. The cycle does not permute the labels (shown as red and blue). (b) Triangulations that differ in the diagonals of a convex pentagon form an *elementary 5-cycle* in the flip graph. This cycle permutes labels as shown in Figure 2.3.

following way. Take a triangulation T and two edges $e, f \in T$ whose removal leaves two internally disjoint convex quadrilaterals in T. Each quadrilateral can be triangulated in two ways, which results in four triangulations that contain $F := T \setminus \{e, f\}$. These four triangulations form a 4-cycle in the flip graph, as shown in Figure 2.4(a). Observe that a traversal of the cycle corresponds to a sequence of flips that returns edge-labels to their original positions.

Define an *elementary* 5*-cycle* to be a cycle of the flip graph obtained in the following way. Take a triangulation T and two edges $e, f \in T$ whose removal leaves a convex pentagon in T. There are five triangulations that contain $F := T \setminus \{e, f\}$, and they form a 5-cycle in the flip graph, as shown in Figure 2.4(b). Observe that the sequence of flips around such a cycle permutes labels of e and f as shown in Figure 2.3.

As a side remark, note that it can be shown that, in fact, any cycle in the flip graph of length less than 6 is an elementary 4- or 5-cycle. However, we will not need this in what follows.

Our main topological theorem is the following.

Theorem 7. *Let P be a set of n points in general position in the plane. There is a high-dimensional cell complex $\mathbb{X} = \mathbb{X}(P)$, which we call the* flip complex*, such that:*

1. *The 1-skeleton of $\mathbb{X}$ is the flip graph of P;*

2. *There is a one-to-one correspondence between the 2-cells of* $\mathbb{X}$ *and the elementary 4-cycles and elementary 5-cycles of the flip graph of* P*;*

3. $\mathbb{X}$ *has the topology of (i.e., is homotopy equivalent to) a high-dimensional ball; therefore its* fundamental group*,* $\pi_1(\mathbb{X})$*, is trivial.*

Theorem 7 follows from a result of Orden and Santos [114], see Remark 15 at the end of Section 2.9.3 below for more details; we are grateful to F. Santos for bringing this reference to our attention.

Before becoming aware of the work of Orden and Santos, we found a different proof of Theorem 7 that starts out by considering the simplicial complex $\mathbb{T} = \mathbb{T}(P)$ whose faces are the sets of pairwise non-crossing edges (line segments) spanned by P. This complex $\mathbb{T}$ is shown to be a *shellable simplicial ball* (by an argument based on constrained Delaunay triangulations), and $\mathbb{X}$ is then constructed as the *dual complex* of $\mathbb{T}$. We hope that this alternative proof of Theorem 7 is of some independent interest and present it in Sections 2.9.2 and 2.9.3 below. Before that, in Section 2.9.1, we show how to derive the Elementary Swap Theorem from Theorem 7.

Some topology references. In what follows, we will use a number of notions from combinatorial topology; most of these we will recall along the way, but others we will only describe informally or leave undefined and instead refer the reader to standard textbooks for further background.

Section 2.9.1 uses only a minimum number of topology notions. For ease of reading, familiarity with a basic definition of a (2-skeleton of a) regular cell complex and of the fundamental group will help, but we introduce most of these in depth, especially the combinatorial definition of the fundamental group. More details on the fundamental group of cell complexes can be found in [133, Chap. 3, 4] or in the further references in the section. A regular cell complex can, intuitively, be thought of as a simplical complex whose faces are no longer required to be simplices but can be any cells (i.e., homeomorphic to balls). A recommended friendly introduction to simplicial complexes is in [104, Chap. 1] and to the cell complexes in [26, Sec. 12] or in [27, Sec. 4.7].

Sections 2.9.2 and 2.9.3 assume some knowledge of (piecewise-linear) topology. Section 2.9.2 discusses properties of simplicial complexes, concretely pureness and a type of face inclusion in complex $\mathbb{T}$ that determine that the complex is a pseudomanifold. The faces that constitute the boundary and the interior of the pseudomanifold will be identified. Crucially, the pseudomanifold will be shown to be shellable (and, consequently, a shellable/piecewice-linear ball). Matoušek's book [104, Chap. 1] gives an excellent overview of basic similarity notions between topological spaces, including homeomorphism, homotopy equivalence or deformation retraction that we use without defining. The other above mentioned topological notions will be defined in the section. More background on shellability can be found in [27, Sec. 4.7] and on piecewise-linear topology of balls, spheres and manifolds in general in [79; 35; 88].

In Section 2.9.3 the dual cell complex $\mathbb{X}$ of the piecewise-linear simplicial ball $\mathbb{T}$ is formed. The important steps and the properties of the dual complex are summarized in Proposition 14 after which the section requires no further topology background. For readers interested in details of the dual construction, the process starts by considering the first barycentric subdivision of $\mathbb{T}$, based on which the decomposition of $\mathbb{T}$ into dual cells is formed. The dual cell complex $\mathbb{X}$ consists of those dual cells that correspond to interior faces of $\mathbb{T}$. See [104; 88] for definition and illustrations of the barycentric subdivision; see Figure 64.1 in [109] for examples of dual cells and see [79, Sec. I.6] or [109, §64 and §70] (which uses the term 'dual blocks' instead of dual cells) for a rigorous description of the construction of dual cells and of the dual cell complex.

2.9.1 From Topology to the Elementary Swap Theorem

In this section we use Theorem 7 to prove the Elementary Swap Theorem. We begin by defining elementary walks. A *walk* in the flip graph is a sequence $T_0, T_1, \dots, T_k$ of triangulations (possibly with repetitions) such that T_{i-1} and T_i differ by a flip. We will refer to T_0 and T_k as the start and the end of the walk, respectively. A walk is *closed* if it starts and ends at the same triangulation. If w_1 and w_2 are walks such that the end of w_1 equals the start of w_2 then we can define their *composition* $w_1 w_2$ in the obvious way. Furthermore, if $w = (T = T_0, T_1, \dots, T_k)$ is a walk, we will use the notation $w^{-1} = (T_k, T_{k-1}, \dots, T_0)$ for the *inverse walk*.

Fix a triangulation T_0. An *elementary quadrilateral walk* is a closed walk of the form

wzw^{-1}, where z is an elementary 4-cycle in the flip graph, and w is a walk from T_0 to some triangulation on z. An *elementary pentagonal walk* is defined analogously, with z an elementary 5-cycle.

It is straightforward to check the effect of these elementary walks on labellings:

Lemma 8. *Let (T_0, ℓ) be a labelled triangulation. An elementary quadrilateral walk from T_0 does not permute the labels. An elementary pentagonal walk swaps the labels of two edges (e and f in Figure 2.4(b)) and leaves all other labels fixed; this corresponds exactly to the notion of an elementary swap introduced earlier.*

Another operation that does not affect the permutation of labels induced by a closed walk is the following. A *spur* ww^{-1} starting and ending at T is an arbitrary walk w starting at T, immediately followed by the inverse walk. If w_1 and w_2 are walks in the flip graph such that w_1 ends at a triangulation T and w_2 starts there, and if s is a spur at T, then we say that the walk $w_1 s w_2$ differs from $w_1 w_2$ by a *spur insertion.* The inverse operation is called a *spur deletion*.

Lemma 9. *If two closed walks w and w' in the flip graph differ only by a finite number of spur insertions and deletions then they yield the same permutation of edge labels.*

Proof. A flip immediately followed by its inverse flip has no effect on labels. The lemma follows by induction on the length of a spur and the number of spur insertions and deletions. □

As proved below, the Elementary Swap Theorem reduces to the following Decomposition Theorem:

Theorem 10. *[Decomposition Theorem / Topological Elementary Swap Theorem] Let w be a closed walk in the flip graph starting and ending at triangulation T_0. Then, up to a finite number of spur insertions and deletions, w can be written as the composition of finitely many elementary walks.*

Proof. [Proof of Theorem 3 (Elementary Swap Theorem), using Theorem 10]

A permutation of labels in a labelled triangulation $\mathcal{T} = (T_0, \ell)$ that can be realized by a sequence of flips corresponds to some closed walk w starting at T_0. The Decomposition Theorem 10 guarantees that, by addition and deletion of finitely many spurs, w

can be expressed as a composition w' of finitely many elementary walks. By Lemma 9, w' causes the same permutation of labels in $\mathcal{T}$ as w. By Lemma 8, w' corresponds to a composition of elementary swaps. $\square$

We prove the Decomposition Theorem 10 using Theorem 7 and the well-known fact that the fundamental group of a regular cell complex can be defined *combinatorially* in terms of closed walks in the 1-skeleton and this definition is equivalent to the usual topological definition in terms of continuous loops. See [26, Sec. 12] or [27, Sec. 4.7] for background on regular cell complexes; see [133, Chap. 3] or [72, Chap. 1] for definition of the fundamental group in terms of continuous loops; and, perhaps most importantly, see [133, Chap. 4] or [127, Chap. 7] for the equivalent combinatorial definition of the fundamental group for regular cell complexes.

We describe the combinatorial definition of the fundamental group of the flip complex $\mathbb{X}$ in detail. The elements of the fundamental group $\pi_1(\mathbb{X})$ are equivalence classes of closed walks in the 1-skeleton of $\mathbb{X}$ with respect to spur insertions, deletions, and the so-called 2-cell operations. Note that the fundamental group of a high-dimensional cell complex is determined by its 2-skeleton, i.e., $\pi_1(\mathbb{X}) = \pi_1(\mathrm{skel}_2(\mathbb{X}))$. (Recall that $\mathrm{skel}_2(\mathbb{X})$ is a subcomplex of $\mathbb{X}$ consisting of its vertices, edges and the 2-cells.)

By Theorem 7, the 1-skeleton of $\mathbb{X}$ is the flip graph of P. Fix a *base triangulation* T_0, and, for every triangulation T, fix a walk p_T from T_0 to T. Given two triangulations T_1, T_2 that differ by a flip, we form the closed walk w_{T_1,T_2} in the flip graph, called a *generating walk*, that goes from T_0 to T_1 along p_{T_1}, then flips to T_2, and then returns to T_0 along $p_{T_2}^{-1}$. It is easy to see that, up to a finite number of spur insertions and deletions, every closed walk starting and ending at T_0 can be written as a composition of generating walks.

We say that walks w and w' are 2*-cell related* if we can express them as $w = w_1w_2$ and $w' = w_1zw_2$, where z is a closed walk traversing the boundary of a 2-cell (an elementary cycle) exactly once in either orientation. See Figure 2.5. Notice that w_1w_2 and $w_1zz^{-1}w_2$ differ only by the spur zz^{-1}, hence, up to spur insertion and deletion, being 2-cell related is symmetric.

Two walks in the flip graph are called *equivalent* if they differ by a finite number of spur insertion and/or deletions and by applying a finite number of 2-cell relations.

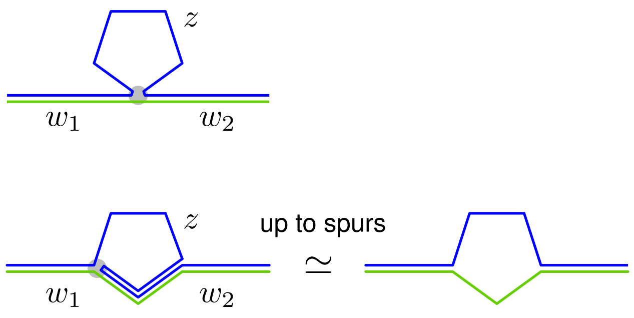


Figure 2.5: Examples of 2-cell related walks: in both examples, the blue walk and the green walk are 2-cell related.

It is not hard to check that this defines an equivalence relation, and the fundamental group $\pi_1(\mathbb{X})$ is defined as the set of equivalence classes of closed walks starting and ending at T_0 (with the group operation being induced by the composition of the closed walks from T_0). Finally, note that since $\mathbb{X}$ is path-connected (i.e., any two points of $\mathbb{X}$ are connected by a path), $\pi_1(\mathbb{X})$ is independent of the choice of the basepoint triangulation T_0.

With this understanding of the combinatorial definition of $\pi_1(\mathbb{X})$, we can prove the Decomposition Theorem.

Proof. [Proof of Theorem 10 (Decomposition Theorem), using Theorem 7]By Theorem 7, the fundamental group of the flip complex $\mathbb{X}$ is trivial and the 1-skeleton of $\mathbb{X}$ is the flip graph. This means that every closed walk w in the flip graph starting at triangulation T_0 is equivalent to the trivial walk, i.e., w and the trivial walk differ by a finite number of spur insertion and/or deletions and by applying a finite number of 2-cell relations. The proof is by induction on the number of 2-cell relations.

Notice the *precomposition property*: whenever two closed walks w'', w' from T_0 are 2-cell related, there is an elementary walk v from T_0 such that, up to spurs, w' can be written as vw''. Indeed, let $w'' = w_1w_2$ and $w' = w_1zw_2$ where z is a closed walk traversing the boundary of a 2-cell. If w'' is precomposed with the closed walk $v = w_1zw_1^{-1}$ then the result $vw'' = (w_1zw_1^{-1})(w_1w_2) = w_1z(w_1^{-1}w_1)w_2$ differs from w' only by the spur $w_1^{-1}w_1$. See Figure 2.6. By Theorem 7, a boundary of a 2-cell is an elementary 4- or 5-cycle and so the walk $v = w_1zw_1^{-1}$ above is indeed an elementary walk.

After applying finitely many 2-cell relations, the original walk w is written, up to

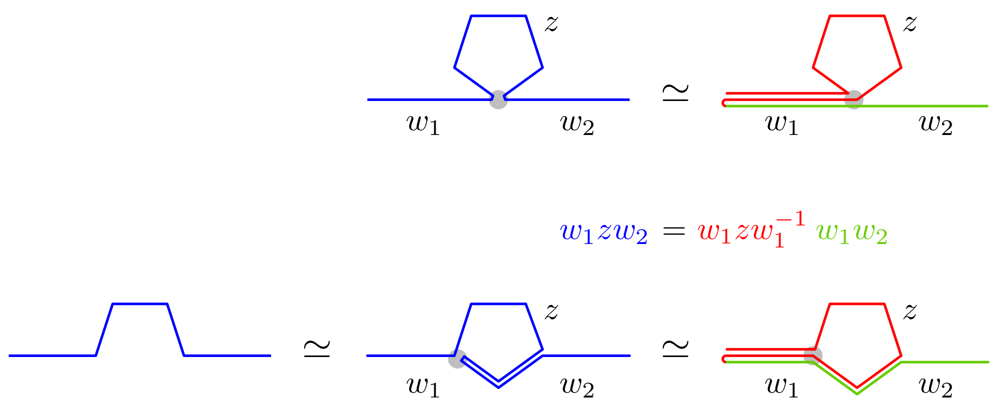


Figure 2.6: Each row gives an example of 2-cell related walks (blue and green), where, up to spurs, the blue walk is written as a composition of an elementary walk (in red) and the green walk.

spurs, as a composition $v_1 \cdots v_k w_{T_0}$ where v_i's are elementary walks and w_{T_0} is the trivial walk from T_0. Hence, up to a finite number of spur insertions and deletions, w is a composition of finitely many elementary walks. $\square$

2.9.2 The Simplicial Complex of Plane Graphs

In this section and the following one, we give a proof of Theorem 7. This section is about the simplicial complex $\mathbb{T} = \mathbb{T}(P)$ whose faces are the sets of pairwise non-crossing edges (line segments) spanned by P.

Let P be a set of n points in general position in the plane, i.e., no three points lie on a line and no four points lie on a common circle. Let E be the set of edges (closed line segments) spanned by P. Two edges $e, f \in E$ are said to be *non-crossing* if they are disjoint or if they intersect in a single point of P that is an endpoint of both edges. We say that a subset $F \subseteq E$ is *non-crossing* if every pair of distinct edges $e, f \in F$ is non-crossing. If G is non-crossing and $F \subseteq G$ then F is non-crossing as well. Thus, the non-crossing sets of edges form an abstract simplicial complex

$$\mathbb{T} = \mathbb{T}(P) := \{F \colon F \subseteq E, F \text{ non-crossing}\},$$

which we call the *complex of plane graphs on P*. We collect some basic properties of $\mathbb{T}$:

1. The *facets* (inclusion-maximal faces) of $\mathbb{T}$ are exactly the triangulations of P (since every non-crossing set of edges $F \subseteq E$ can be extended to a triangulation). Thus, the simplicial complex $\mathbb{T}$ is of dimension $m-1$, where m is the number of edges in any triangulation of P, and it is *pure*, i.e., every face of $\mathbb{T}$ is contained in a face of dimension $m-1$.

2. Every face F of $\mathbb{T}$ of dimension $m-2$ is contained in either one or two triangulations. In the latter case, F corresponds to a flip between these two triangulations.

We will show that the topology of $\mathbb{T}$ is particularly simple, namely that $\mathbb{T}$ is homeomorphic to an $(m-1)$-dimensional ball. Furthermore, there is a combinatorial certificate (*shellability*) for this homeomorphism. This implies that the homeomorphism is particularly nice and that $\mathbb{T}$ is a *piecewise-linear ball*, which means that there is a subdivision $\mathbb{T}'$ of $\mathbb{T}$ such that $\mathbb{T}'$ is simplicially isomorphic to a subdivision $\mathbb{B}'$ of the d-dimensional simplex Δ^d. only property of piecewise-linearity that we will need is that it ensures that the construction of the *dual cell complex* $\mathbb{T}^*$ is well-behaved (see Proposition 14 below).

We refer to [27, Sec. 4.7] and [79; 35; 88] for more details and further references on shellability and piecewise-linear balls, spheres, and manifolds.

We recall that a pure d-dimensional simplicial complex $\mathbb{K}$ is *shellable* if there exists a total ordering of its facets $F_1, F_2, \cdots, F_N$ (called a *shelling order*) such that, for every $2 \leq j \leq N$, the intersection of F_j with the simplicial complex generated by the preceding facets is pure of dimension $d-1$, i.e., for every $i < j$ and $F := F_i \cap F_j$, there exists some $k < j$ such that $G := F_j \cap F_k$ is of dimension $d-1$ and $F \subseteq G$.

We will need the following result (which appears implicitly in [23], and explicitly in [46]; see [27, Prop. 4.7.22] for a short proof).

Proposition 11. *Suppose $\mathbb{K}$ is a finite d-dimensional simplicial complex that is a pseudomanifold, i.e., $\mathbb{K}$ is pure and every $(d-1)$-dimensional face of $\mathbb{K}$ is contained in at most two d-faces. If $\mathbb{K}$ is shellable then $\mathbb{K}$ is either a piecewise-linear ball or a piecewise-linear sphere. The former case occurs iff there is at least one $(d-1)$-dimensional face that is contained in only one d-face of $\mathbb{K}$, in which case the pseudomanifold is said to have non-empty boundary.*

We remark that the property of being a shellable pseudomanifold (which is a combinatorial and algorithmically verifiable condition) is strictly stronger than being a piecewise-linear ball or sphere, which in turn is strictly stronger than being a simplicial complex homeomorphic to a ball or sphere.

Using Proposition 11, we now prove:

Theorem 12. *$\mathbb{T}$ is a shellable $(m-1)$-dimensional pseudomanifold with non-empty boundary, and hence a piecewise-linear ball.*

Proof. We observed earlier that $\mathbb{T}$ is a pure $(m-1)$-dimensional simplicial complex, and that every $(m-2)$-dimensional face of $\mathbb{T}$ is contained in at most two $(m-1)$-dimensional faces, hence $\mathbb{T}$ is a pseudomanifold. Moreover, if T is a triangulation of P and if $e \in T$ is a non-flippable edge (e.g., if e is a convex hull edge) then $F := T \setminus \{e\}$ is an $(m-2)$-dimensional face of $\mathbb{T}$ that is contained in a unique $(m-1)$-face, namely T. Hence, $\mathbb{T}$ has non-empty boundary.

Thus, by Proposition 11, it suffices to show that $\mathbb{T}$ is shellable, i.e., to exhibit a shelling order for the facets of $\mathbb{T}$.

With every triangulation T of P, we associate the sorted vector of angles $\alpha(T) = (\alpha_1(T), \alpha_2(T), \cdots, \alpha_{3t}(T))$, where $\alpha_1(T) \leq \alpha_2(T) \leq \cdots \leq \alpha_{3t}(T)$ are the angles occurring in the triangulation T. We order the triangulations of P by sorting the corresponding angle vectors $\alpha(T)$ lexicographically from largest to smallest. Since we assume P to be in general position, this defines a total ordering of triangulations of P,

$$T_1, T_2, \ldots, T_N, \qquad \alpha(T_1) >_{\text{LEX}} \alpha(T_2) >_{\text{LEX}} \cdots >_{\text{LEX}} \alpha(T_N), \tag{2.1}$$

where N is the number of triangulations of P.

It is well known (see, for example, [50, Chap. 3.4]) that in this ordering, T_1 is the Delaunay triangulation of P. Moreover, if we consider only triangulations containing a particular plane subgraph corresponding to a face F of $\mathbb{T}$ and the corresponding subsequence of the angle vectors, the first of these vectors corresponds to the Delaunay triangulation constrained to F.

We claim that the triangulation ordering (2.1) defines a shelling. We need to prove that for every $i < j \leq N$ and $F := T_i \cap T_j$, there exists some $k < j$ such that $G := T_k \cap T_j$

is of dimension $m-2$ and $F \subseteq G$.

To see this, consider the subsequence $T_{k_1}, T_{k_2}, \ldots$ of the sequence (2.1) consisting only of those triangulations that contain the edge set F. Then T_{k_1} is the constrained Delaunay triangulation with respect to the edge set F, and T_i and T_j both appear in that subsequence; in particular, $T_j \neq T_{k_1}$ since T_i precedes it. Since every triangulation containing F can be transformed to the constrained Delaunay triangulation T_{k_1}, (see, e.g., the description of Lawson's flip algorithm in [50]) there must exist an edge $e \in T_j \setminus T_{k_1}$ such that flipping e (a Lawson flip) increases the angle vector; thus, the triangulation resulting from flipping e is some T_k with $k < j$ and satisfies $F \subseteq T_k \cap T_j$ as desired. □

Next, we need a characterization of interior versus boundary faces of $\mathbb{T}$. Let $\mathbb{B}$ be a piecewise-linear ball of dimension d. By definition, the *boundary* $\partial\mathbb{B}$ of $\mathbb{B}$ is the subcomplex of $\mathbb{B}$ consisting of all faces F for which there exists a $(d-1)$-dimensional face G of $\mathbb{B}$, with $F \subseteq G$, such that G is contained in a unique d-dimensional face of $\mathbb{B}$, see, for example, [35; 88]. (In the case $\mathbb{B} = \mathbb{T}$, the latter condition means that $G = T \setminus \{e\}$ for some triangulation T and some edge $e \in T$ that is not flippable.) A face F of $\mathbb{B}$ that does not lie in $\partial\mathbb{B}$ is called an *interior face*.

For the proof of Theorem 7 we need properties of interior faces of $\mathbb{T}$ of dimensions $m-1$, $m-2$ and $m-3$. The following proposition characterizes interior faces more generally.

Proposition 13. *Let $\mathbb{T}$ be the simplicial complex of plane graphs on the point set P. A non-crossing set of edges F on P is an interior face of $\mathbb{T}$ if and only if the following conditions hold:*

(i) F contains all convex hull edges of P;

(ii) Every bounded region in the complement of the plane graph (P, F) is convex.

Proof. Note that a polygon is non-convex iff it has a reflex vertex. More generally, a bounded region in the complement of the plane graph (P, F) is non-convex iff there is an interior point p of P and a half-plane H through p with no edge of F from p to a point interior to H—in this case we say that p "has no edge in a half-plane". The statement of the proposition is then equivalent to the following: F is a boundary face if and only if F misses a convex hull edge or there is an interior point p of P with no edge in a half-plane. We prove this statement.

For the forward direction, suppose that F is a boundary face. Then there is a triangulation T, $F \subseteq T$, and an edge $e \in T - F$ such that e is not flippable in T. If e is a convex hull edge, then F does not contain all convex hull edges. Otherwise e is a diagonal of a non-convex quadrilateral in T. Set p to be the reflex vertex of the non-convex quadrilateral and H to contain the other end of e but not the two other vertices of the quadrilateral. Then p has no edge in the half-plane H.

For the other direction, first note that if F misses a convex hull edge then F is a boundary face. For the other case, suppose that in F there is a non-convex hull point p of P that has no edge in the half-plane H. Augment F to a maximal set F' of non-crossing edges without using any edge from p into H. This will not yet be a triangulation (because in a triangulation p is surrounded by triangles and they have angles bounded by π). Now augment F' further to a triangulation T. Then $T - F'$ contains some edge e incident to p, and e is not flippable otherwise we could have further augmented F'. Thus F is a boundary face. $\square$

2.9.3 The Dual Flip Complex $\mathbb{X}$

In this section we define the dual flip complex and prove Theorem 7.

To define the flip complex $\mathbb{X}$, we need the notion of *dual cells* and the dual cell decomposition of a piecewise-linear ball; for the precise definition, we refer to [79, Sec. I.6] or [109, §64 and §70].

In [109], the terminology *dual blocks* is used instead of dual cells, since the construction is described in a more general setting (for arbitrary triangulated manifolds or homology manifolds) in which the dual blocks F^* need not be cells (homeomorphic to balls). However, in the setting of piecewise-linear manifolds, in particular of piecewise-linear balls, as in our case, this technical issue does not arise.

Here, we simply collect the properties that we will need:

Proposition 14. *Let $\mathbb{B}$ be a d-dimensional piecewise-linear ball.*

1. For each interior k-dimensional face F of $\mathbb{B}$, one can define a *dual cell* F^* (a certain subcomplex of the *barycentric subdivision* of $\mathbb{B}$ that is a piecewise-linear ball of dimension $d-k$ [79, Lemma 1.19]).
2. The construction reverses inclusion, i.e., for interior faces F, G of $\mathbb{B}$, $F \subseteq G$ iff $F^* \supseteq G^*$.
3. The dual cells of the interior faces of $\mathbb{B}$ form a regular cell complex, denoted $\mathbb{B}^*$ and called the *dual cell complex*. $\mathbb{B}^*$ need not be a manifold or pure d-dimensional, but it is homotopy equivalent to $\mathbb{B}$ [109, Lem. 70.1]. (Technically, the dual complex of a piecewise-linear manifold with boundary is a deformation retraction of the manifold. For manifolds without boundary, the dual complex is piecewise-linearly homeomorphic to the original manifold.)

We define the *flip complex* $\mathbb{X} := \mathbb{T}^*$ as the dual complex of the simplicial complex $\mathbb{T}$.

Proof of Theorem 7. By Theorem 12, $\mathbb{T}$ is a piecewise-linear ball and thus it has a trivial fundamental group. By Proposition 14, $\mathbb{X} = \mathbb{T}^*$ is a regular cell complex that is homotopy equivalent to the ball $\mathbb{T}$. Consequently (using the fact that homotopy equivalent spaces have isomorphic fundamental groups, see, for example, [72]), the fundamental group $\pi_1(\mathbb{X})$ is trivial.

It remains to show the characterization of the vertices, edges, and 2-cells of $\mathbb{X}$.

The vertices of $\mathbb{X}$ correspond (are dual) to the faces of $\mathbb{T}$ of the highest dimension $(m-1) = \dim \mathbb{T}$, i.e., to the triangulations of P (these are automatically interior faces of $\mathbb{T}$).

The edges of $\mathbb{X}$ correspond to interior $(m-2)$-dimensional faces F of $\mathbb{T}$, i.e., faces F that are contained in two triangulations T and T' that differ by a flip. Thus, the 1-skeleton of $\mathbb{X}$ is exactly the flip graph of P.

Every 2-cell of $\mathbb{X}$ is the dual cell F^* of an interior face F of $\mathbb{T}$ of dimension $m-3 = \dim F$. Consider an arbitrary triangulation T containing F, i.e., F is obtained from T by deleting two edges e, f. By Proposition 13, e and f are both flippable in T since they lie in a convex polygon in T.

If e and f are not incident to a common triangle in T, (or, equivalently, removing both e and f from T creates two internally disjoint convex quadrilaterals) then there exist four triangulations containing F and these form an elementary 4-cycle in the flip graph. It follows from the definition of $\mathbb{X} = \mathbb{T}^*$ that the 4-cycle is the boundary of the dual cell F^*.

Otherwise, e and f are incident to a common triangle in T. By Proposition 13 the union of the three triangles of T containing either e or f forms a convex polygon, necessarily a pentagon. There are five triangulations containing F and these form an elementary 5-cycle in the flip graph. It follows from the definition of $\mathbb{X} = \mathbb{T}^*$ that the 5-cycle is the boundary of the dual cell F^*.

Hence, every 2-cell of $\mathbb{X}$ corresponds to an elementary 4- or 5-cycle of the flip graph.

Conversely, every elementary 4- or 5-cycle of the flip graph gives rise to a 2-cell F^* of $\mathbb{X}$: more precisely, F^* corresponds to the intersection of the triangulations in the elementary cycle. □

Remark 15. As remarked above, the flip complex $\mathbb{X}$ and Theorem 7 are closely related to a result of Orden and Santos [114]. Specifically, Orden and Santos showed that for every point set P, there exist a simple polytope $\mathbb{Y} = \mathbb{Y}(P)$ and a distinguished face F_0 of $\mathbb{Y}$ with the following properties: The vertices of F_0 correspond to *pseudotriangulations* with vertex set P that are not triangulations. Recall that a pseudotriangulation of P is a decomposition of the convex hull of P into *pseudotriangles*, i.e., possibly non-convex polygons with exactly three non-reflex vertices. By contrast, the vertices of $\mathbb{Y}$ that do *not* lie in the distinguished face F_0 are in one-to-one correspondence with the triangulations of P. More generally, the faces of $\mathbb{Y}$ that are disjoint from the distinguished face F_0 are in one-to-one correspondence with the non-crossing sets of edges of P that contain all convex hull edges of P. Furthermore, the correspondence reverses inclusion.

It follows from this that the cell complex $\mathbb{K} = \mathbb{K}(P)$ of all faces of $\mathbb{Y}$ disjoint from F_0 has the flip graph of P as its 1-skeleton, and the fundamental group of $\mathbb{K}$ is trivial (since $\mathbb{K}$ is the complement of the star of a face in the boundary of a convex polytope, where a star of face F_0 consists of all faces of $\mathbb{Y}$ that have a non-empty intersection with F_0); analogously to the proof of Theorem 7, it can be shown that the 2-faces of $\mathbb{K}$ correspond to the elementary 4-cycles and 5-cycles in the flip graph. Thus, the complex $\mathbb{K}$ could be used instead of the flip complex $\mathbb{X}$ to prove the Elementary Swap

Theorem.

A different way of viewing the complex $\mathbb{K}$ is as follows: Let C be the set of convex hull edges of P. Then C is a face of the complex $\mathbb{T}$ of plane graphs on P, and since $\mathbb{T}$ is a shellable ball, the link $\mathbb{L}$ of C in $\mathbb{T}$ is a shellable (and hence piecewise-linear) ball or sphere (see page 84 for a definition of a link of a simplex). The complex $\mathbb{K}$ of Orden–Santos is the dual cell complex of $\mathbb{L}$.

2.10 Proofs of Properties of Elementary Swaps

In this section we prove Lemmas 4 and 5. For ease of reading, we repeat the Lemmas here:

Lemma 4. *If there is an elementary swap between two edges in a triangulation $\mathcal{T}$ then there is a flip sequence of length $O(n^6)$ to realize the elementary swap, and, furthermore, this sequence can be found in polynomial time.*

Lemma 5. *Let $\mathcal{T}$ be a labelled triangulation containing two edges e and f. If there is a sequence of elementary swaps on $\mathcal{T}$ that takes the label of edge e to edge f, then there is an elementary swap of e and f in $\mathcal{T}$.*

To prove Lemma 4, the idea is to look at paths in the *double quadrilateral graph* G_D that we will define below. Informally speaking, G_D captures where pairs of non-crossing edges can go via flips, similar to the way the quadrilateral graph captures where a single edge can go via flips. We will show that there is an elementary swap between two labels in a triangulation if and only if there exists a path of certain type in the double quadrilateral graph.

Proof of Lemma 4. Construct a graph G_D called the *double quadrilateral graph*. Vertices of the graph G_D are pairs of non-crossing edges on the point set P, and we define two vertices (e_1, f_1) and (e_2, f_2) of G_D to be adjacent if either $e_1 = e_2$ and f_1 and f_2 are adjacent in the quadrilateral graph, or if $f_1 = f_2$ and e_1 and e_2 are adjacent in the quadrilateral graph. (Recall that two edges a and b are adjacent in the quadrilateral graph if a and b cross and their four endpoints form an empty quadrilateral.)

In the graph G_D we identify some vertices as “swap vertices”. These are the vertices (g, h) such that g and h are diagonals of some empty convex pentagon in the point set. Note that the swap vertices can be identified in polynomial time.

We claim that there is an elementary swap of e and f in the labelled triangulation $\mathcal{T} = (T, \ell)$ if and only if there is a path in G_D from vertex (e, f) to a swap vertex. For the forward direction, suppose there is such an elementary swap. It begins with a sequence σ of flips from $\mathcal{T}$ to a labelled triangulation $\mathcal{T}'$ in which labels $\ell(e)$ and $\ell(f)$ are attached to two diagonals g and h of some empty convex pentagon. The subsequence of σ consisting of those flips that apply to an edge whose current label is $\ell(e)$ or $\ell(f)$ corresponds to a path in G_D from (e, f) to the swap vertex (g, h).

For the other direction, let π be a path in G_D from (e, f) to a swap vertex. It suffices to show that the path π provides a sequence of flips, σ, that takes $\mathcal{T}$ to some labelled triangulation $\mathcal{T}'$ in which labels $\ell(e)$ and $\ell(f)$ are attached to two diagonals of an empty convex pentagon, because the rest of the elementary swap is then determined. Consider the first edge of π and suppose without loss of generality that it goes from (e, f) to (e, f') (the case when e changes is similar). Then e and f' are non-crossing (by definition of vertices in G_D). Because f and f' are adjacent in the quadrilateral graph, they cross and form an empty convex quadrilateral Q. Note that e does not intersect the interior of Q, since Q is empty and e does not cross f or f'. We apply the result that any constrained triangulation can be flipped to any other with $O(n^2)$ flips. Constrain edges e and f in T and flip $\mathcal{T}$ to a labelled triangulation that contains the edges of quadrilateral Q. In this triangulation, we can flip f to f', transferring $\ell(f)$ to f'. We continue in this way to realize each edge of π via $O(n^2)$ flips, arriving finally at a labelled triangulation in which labels $\ell(e)$ and $\ell(f)$ are attached to edges that are the diagonals of some empty convex pentagon in the point set. Fixing the two diagonals, we can flip to a triangulation that contains the edges of the convex pentagon, and at this point we are done.

Because the graph G_D has $O(n^4)$ vertices, the diameter of any of its connected components is $O(n^4)$. Thus, if there is an elementary swap that exchanges the labels of edges e and f, then there is one corresponding to a path in G_D of length $O(n^4)$. We can explicitly construct G_D and find a path between (e, f) and a swap vertex in G_D in polynomial time. As argued above, every edge of G_D can be realized by $O(n^2)$ flips.

This proves that, for any elementary swap, we can construct a sequence of $O(n^6)$ flips to realize it, and the construction takes polynomial time. $\square$

As mentioned in Section 2.8, there is a group-theoretic argument proving a weaker version of Lemma 5. The argument depends on the following claim: If a permutation group is generated by transpositions and contains a permutation that maps element e to f then the group contains the transposition of e and f. To prove this claim, notice that if the group contains transpositions (ab) and (bc), then it also contains transposition $(ac) = (ab)(bc)(ab)$; and apply induction.

To apply this claim in our situation, observe that by the Elementary Swap Theorem, all label permutations achievable by flips in a triangulation $\mathcal{T}$ are compositions of elementary swaps, hence, these label permutations indeed form a group G generated by transpositions. Moreover, by the assumption of Lemma 5, G contains a permutation taking the label of edge e to edge f. Hence, by the above claim, the group G also contains a permutation, which is a composition of elementary swaps, whose effect is to transpose labels of edges e and f.

In order to prove the full result of Lemma 5, i.e., that the label transposition of e and f can be done with a single elementary swap, we combine the techniques used in the proof of the group theory claim above with the structure of elementary swaps.

Proof of Lemma 5. An elementary swap in triangulation $\mathcal{T}$ acts on two edges of $\mathcal{T}$. We define a graph G_S called the *elementary swap graph* of $\mathcal{T}$. G_S has a vertex for every edge of $\mathcal{T}$, and we define vertices e and f to be adjacent in G_S if there is an elementary swap of e and f in $\mathcal{T}$.

By hypothesis, there is a sequence of elementary swaps that takes the label of edge e to edge f. Observe that no sequence of elementary swaps will take the label of edge e outside the connected component of G_S that contains e. Therefore e and f must lie in the same connected component of G_S. We will now show that each connected component of G_S is a clique. This implies that there is an elementary swap of e and f, and completes our proof.

Consider a simple path $(e_0, e_1), (e_1, e_2), \dots, (e_{k-1}, e_k)$ in G_S. Let σ_i, $i = 1, \dots, k$ be a flip sequence that realizes the elementary swap (e_{i-1}, e_i), and let $\sigma = \sigma_1\sigma_2\dots\sigma_{k-1}$. Observe that σ takes the label of e_0 to e_{k-1}, and does not change the label of e_k (by the assumption that the path is simple). By definition of an elementary swap, the flip sequence σ_k has the form $\rho\pi\rho^{-1}$ where ρ is a sequence of flips that moves the labels of e_{k-1} and e_k into an empty convex pentagon, and π is the sequence of five flips that exchanges the labels of e_{k-1} and e_k.

Consider the flip sequence $\sigma\sigma_k\sigma^{-1} = \sigma\rho\pi\rho^{-1}\sigma^{-1} = \sigma\rho\pi(\sigma\rho)^{-1}$. The first part of this flip sequence, $\sigma\rho$, moves the labels of e_0 and e_k into an empty convex pentagon; the middle part, π, exchanges them; and the final part, $(\sigma\rho)^{-1}$ reverses the first part. Therefore this flip sequence realizes an elementary swap of e_0 and e_k. □

By this we concluded the proof of the Orbit Theorem 1.

A summary sketch of the proof of the Orbit Theorem from Sections 2.4 – 2.10 is displayed in Figure 2.7.

Finally, we include a summary of the polynomial-time algorithm that checks whether reconfiguration between two labelled triangulations $\mathcal{T}_1 = (T_1, l_1)$ and $\mathcal{T}_2 = (T_2, l_2)$ is possible and if so, finds a flip sequence of length $O(n^7)$ between them:

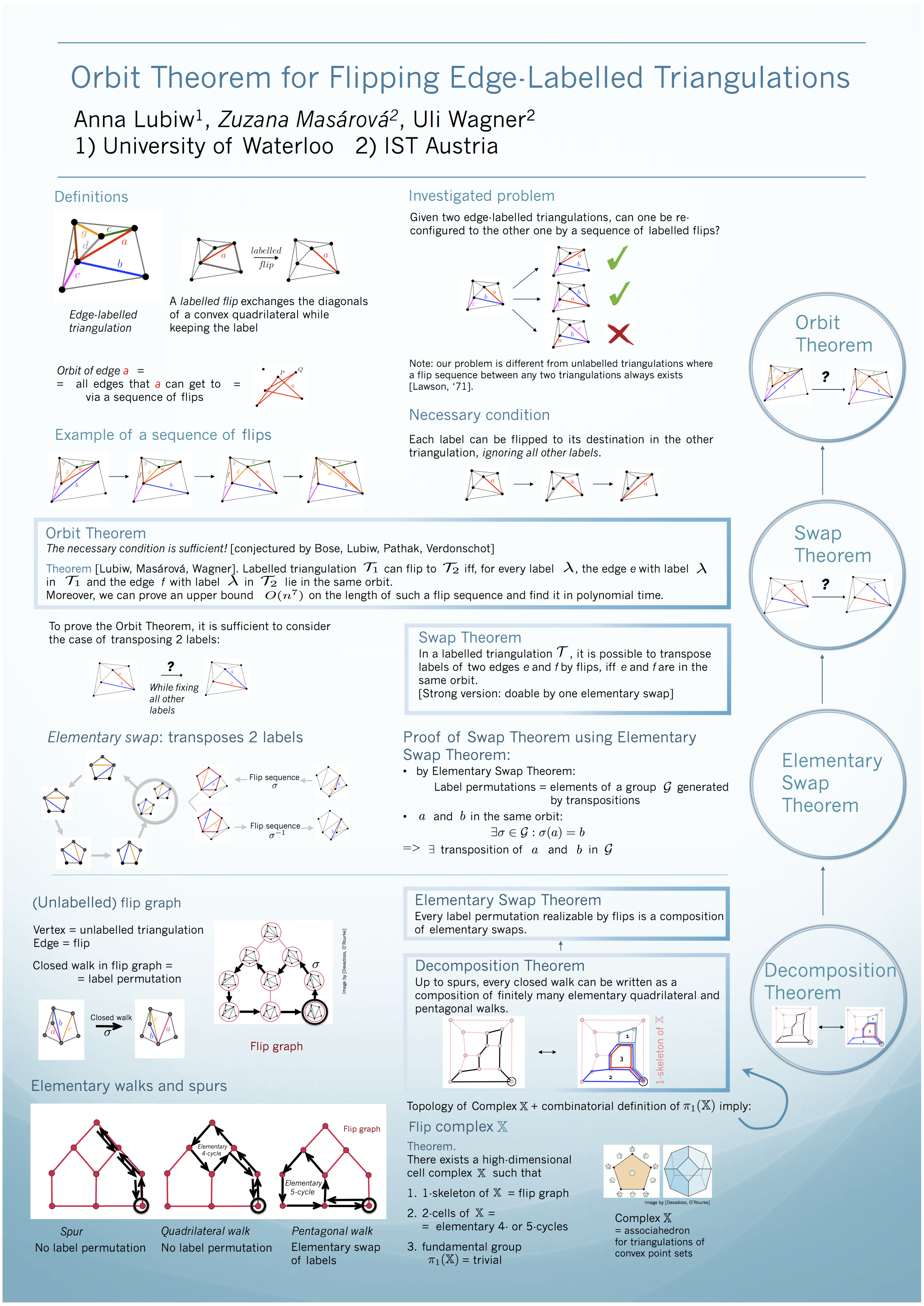
1. Given the two labelled triangulations, find orbits of the point set and determine whether for each label l, the edges in $\mathcal{T}_1$ and $\mathcal{T}_2$ having label l belong to the same orbit.
   If so, a flip sequence reconfiguring $\mathcal{T}_1$ into $\mathcal{T}_2$ can be found by the steps 2–3.
   Otherwise, reconfiguration between $\mathcal{T}_1$ and $\mathcal{T}_2$ is not possible.

2. Flip triangulation $\mathcal{T}_1$ into a labelled triangulation $\mathcal{T}' = (T', l')$ where $T' = T_2$. Note that edges having the same label in $\mathcal{T}'$ and $\mathcal{T}_2$ belong to the same orbit.

3. Effect the label permutation between $\mathcal{T}'$ and $\mathcal{T}_2$ by moving one label at a time to its target position by a single elementary swap. To find an elementary swap of length $O(n^6)$ between two edges e and f in triangulation $\mathcal{T}'$:

   (a) construct the double quadrilateral graph G_D of the point set. Identify its "swap" vertices (as defined in the proof of Lemma 4), and the vertex (e, f) corresponding to the pair of edges e, f.

   (b) Find a path between (e, f) and some swap vertex in G_D and convert the path to a flip sequence (as in Lemma 4) of length $O(n^6)$ that carries out an elementary swap of edges e and f.



Orbit Theorem for Flipping Edge-Labelled Triangulations

Anna Lubiw[1], *Zuzana Masárová*[2], Uli Wagner[2]
1) University of Waterloo 2) IST Austria

Definitions

Edge-labelled triangulation

A *labelled flip* exchanges the diagonals of a convex quadrilateral while keeping the label

Orbit of edge a =
= all edges that a can get to via a sequence of flips =

Example of a sequence of flips

Investigated problem

Given two edge-labelled triangulations, can one be re-configured to the other one by a sequence of labelled flips?

Note: our problem is different from unlabelled triangulations where a flip sequence between any two triangulations always exists [Lawson, '71].

Necessary condition

Each label can be flipped to its destination in the other triangulation, *ignoring all other labels.*

Orbit Theorem

The necessary condition is sufficient! [conjectured by Bose, Lubiw, Pathak, Verdonschot]

Theorem [Lubiw, Masárová, Wagner]. Labelled triangulation $\mathcal{T}_1$ can flip to $\mathcal{T}_2$ iff, for every label λ, the edge e with label λ in $\mathcal{T}_1$ and the edge f with label λ in $\mathcal{T}_2$ lie in the same orbit.
Moreover, we can prove an upper bound $O(n^7)$ on the length of such a flip sequence and find it in polynomial time.

To prove the Orbit Theorem, it is sufficient to consider the case of transposing 2 labels:

While fixing all other labels

Swap Theorem

In a labelled triangulation $\mathcal{T}$, it is possible to transpose labels of two edges e and f by flips, iff e and f are in the same orbit.
[Strong version: doable by one elementary swap]

Elementary swap: transposes 2 labels

Flip sequence σ

Flip sequence σ^{-1}

Proof of Swap Theorem using Elementary Swap Theorem:

- by Elementary Swap Theorem:
  Label permutations = elements of a group $\mathcal{G}$ generated by transpositions
- a and b in the same orbit:
  $\exists \sigma \in \mathcal{G} : \sigma(a) = b$

=> $\exists$ transposition of a and b in $\mathcal{G}$

(Unlabelled) flip graph

Vertex = unlabelled triangulation
Edge = flip

Closed walk in flip graph =
= label permutation

Closed walk σ

Flip graph

Image by [Devadoss, O'Rourke]

Elementary walks and spurs

Spur	*Quadrilateral walk*	*Pentagonal walk*
No label permutation	No label permutation	Elementary swap of labels

Elementary Swap Theorem

Every label permutation realizable by flips is a composition of elementary swaps.

Decomposition Theorem

Up to spurs, every closed walk can be written as a composition of finitely many elementary quadrilateral and pentagonal walks.

Topology of Complex $\mathbb{X}$ + combinatorial definition of $\pi_1(\mathbb{X})$ imply:

Flip complex $\mathbb{X}$

Theorem.
There exists a high-dimensional cell complex $\mathbb{X}$ such that

1. 1-skeleton of $\mathbb{X}$ = flip graph
2. 2-cells of $\mathbb{X}$ =
   = elementary 4- or 5-cycles
3. fundamental group
   $\pi_1(\mathbb{X})$ = trivial

Image by [Devadoss, O'Rourke]

Complex $\mathbb{X}$
= associahedron for triangulations of convex point sets

Figure 2.7: Summary of the proof of the Orbit Theorem from Sections 2.4 – 2.10.

2.11 Orbit Theorem for Constrained Triangulations: existence of a flip sequence respecting some fixed edges

In this section we prove the Orbit Theorem for constrained edge-labelled triangulations. In particular, we show that the theorem holds if a subset of edges in a triangulation of point set has to stay fixed during reconfiguration. It will follow that the Orbit Theorem also holds for triangulations of simple polygons.

Let S be a set of pairwise non-crossing edges on a point set P; we call S the *fixed edges*. Let e and f be edges on P such that $e \cup S$ and $f \cup S$ are both non-crossing. We say that the edges e and f lie in the same *orbit with respect to* S if we can attach label l to e in some triangulation containing S and apply some sequence of flips, never flipping any edge of S, to arrive at a triangulation in which edge f has label l.

In this section we prove the following theorem:

Theorem 16 (Orbit Theorem for Constrained Triangulations)**.** *Given two edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ of a point set, and a set S of fixed edges that apear in both $\mathcal{T}_1$ and $\mathcal{T}_2$, there is a flip sequence that transforms one triangulation into the other, while keeping edges of S fixed, if and only if for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_2$ having label l belong to the same orbit with respect to S.*

Furthermore, there is a polynomial-time algorithm (with $O(n^8)$ being a crude bound on its run-time) that tests whether the condition is satisfied, and if it is, computes a flip sequence of length $O(n^7)$ to transform $\mathcal{T}_1$ to $\mathcal{T}_2$.

Without loss of generality we will assume that S does not contain any convex hull edges of the point set.

Under the stated conditions, Theorem 16 claims the existence of a relatively short reconfiguration sequence. In Section 2.12 we will show that, however, as compared to the case when all edges can be flipped, in the constrained case the length of a shortest flip sequence may increase. Section 2.12 will provide results analogous to a token swapping reconfiguration problem that we study in Chapter 3.

In the rest of this section we prove Theorem 16 by adapting the original proof of the Orbit Theorem 1 to the constrained case. Throughout we refer to the relevant proofs from Sections 2.6 - 2.10.

The key is to prove the constrained analogue of the Decomposition Theorem 10, after which the rest of the proof carries over to the constrained case with only minor modifications.

The flip graph constrained to the set of fixed edges S is smaller and contains fewer elementary 4- and 5-cycles than the full flip graph on the point set. Nevertheless we will be able to prove a Constrained Decomposition Theorem 19 showing that any label permutation that can be realized by a closed walk in the constrained flip graph can also be realized by a composition of finitely many elementary walks in that flip graph. The elementary 4- and 5-cycles and the elementary walks are defined as in Section 2.9 (except now the walks are being performed in the constrained flip graph). As before, there are two types of elementary walks. The elementary quadrilateral walks result in trivial label permutations and elementary pentagonal walks result in elementary swaps of labels, respecting edges of S. Thus the Elementary Swap Theorem for the contrained case immediately follows from the Constrained Decomposition Theorem 19 by the same argument as in the proof of Theorem 3 on page 67:

Theorem 17 (Elementary Swap Theorem for Constrained Triangulations)**.** Given a labelled triangualtion $\mathcal{T}$ and a set S of its edges, any permutation of the labels that can be realized by a sequence of flips fixing the edges of S can be realized by a sequence of elementary swaps fixing the edges of S.

By the same argument as before, from the Constrained Elementary Swap Theorem one proves the Constrained Swap Theorem, see the proof of Theorem 2 on page 61:

Theorem 18 (Swap Theorem for Constrained Triangulations)**.** In a labelled triangulation $\mathcal{T}$, two edges are in the same orbit with respect to S if and only if there is an elementary swap between them that fixes the edges of S.

Finally, to find a flip sequence between the two given edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$, transform (by flips respecting S) the triangulation $\mathcal{T}_1$ into edge-labelled triangulation $\mathcal{T}_1'$ so that $\mathcal{T}_1'$ and $\mathcal{T}_2$ share the same underlying unlabelled triangulation. Lastly, build the corresponding label permutation between $\mathcal{T}_1'$ and $\mathcal{T}_2$ from elementary swaps.

As before, each elementary swap can be realized by a flip sequence of length $O(n^6)$ which can be found in polynomial time. This follows from the same argument as in the proof of Lemma 4 on page 77, only using the double quadrilateral graph $G_D|S$ restricted with respect to the set S: vertices of the graph $G_D|S$ are pairs of edges e, f on the point set P such that $e, f \notin S$ and $S \cup e \cup f$ is non-crossing. The vertex adjacency in the graph $G_D|S$ is defined as in the original double quadrilateral graph.

The rest of the proof, including the algorithm on page 80 that computes a reconfiguration sequence if it exists, carries through (only this time using orbits with respect to S instead of the original orbits). We conclude that the flip sequence between triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$, if it exists, can be computed in polynomial time and so that it consists of $O(n^7)$ flips.

Thus, the only missing piece to prove the Constrained Orbit Theorem 16 is to prove the Decomposition Theorem for closed walks in the flip graph constrained to S:

Theorem 19. *[Decomposition Theorem for Constrained Triangulations] Let w be a closed walk in the (unlabelled) flip graph constrained to S that starts and ends at triangulation T_0. Then, up to a finite number of spur insertions and deletions, w can be written as a composition of finitely many elementary walks in the constrained flip graph.*

To prove the Constrained Decomposition Theorem 19, we will use a constrained flip complex $\mathbb{X}_S$ defined as the dual complex to a certain subcomplex $\mathbb{T}_S$ of the original complex of plane graphs $\mathbb{T}$. Faces of $\mathbb{T}_S$ will correspond to plane graphs F, not containing any of the edges in S and such that $F \cup S$ is non-crossing. The constrained flip complex $\mathbb{X}_S$ will be shown to have a trivial fundamental group, to have the flip graph constrained to S as its 1-skeleton, and the elementary 4- and 5-cycles of the constrained flip graph corresponding to its 2-cells.

To define the simplicial complex $\mathbb{T}_S$, we start by considering links. The *link* of a simplex σ in a simplicial complex X, denoted by $lk(\sigma)$, is defined as follows:

$$lk(\sigma) := \{\tau \in X : \tau \cap \sigma = \emptyset \text{ and } \tau \cup \sigma \in X\},$$

i.e., the link of σ consists of such simplices τ in X which are (as abstract simplices) disjoint from σ and for which there is a simplex in X that contains both σ and τ as its face.

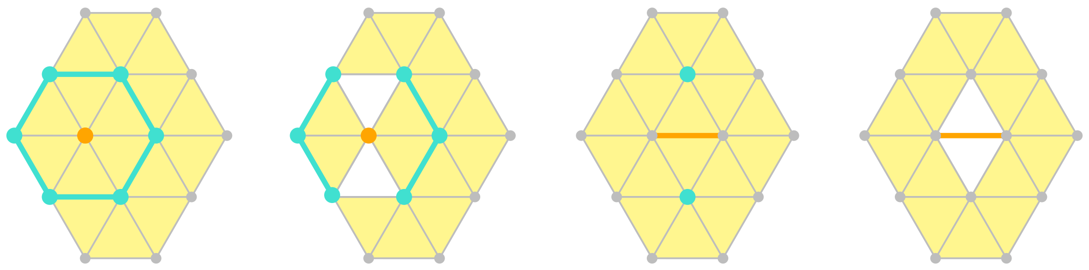


Figure 2.8: Examples of links. The entire simplicial complex consists of all vertices, edges and yellow triangles. The white triangles are holes. Simplex σ is in orange, in the first two examples σ is a vertex, in the last two σ is an edge. The corresponding links $lk(\sigma)$ are in green (the last link is empty).

Note that $lk(\sigma)$ is a subcomplex of X: indeed, $lk(\sigma)$ is downward-closed since for any simplex $\tau' \subseteq \tau$, we have $\tau' \cap \sigma = \emptyset$ and $\tau' \cup \sigma \in X$ since X is downward-closed.

Recall that, as defined in the proof of the Orbit Theorem, $\mathbb{T}$ is a simplicial complex of pairwise non-crossing line segments on the point set P. In particular, the vertices of $\mathbb{T}$ correspond to single segments on P, edges of $\mathbb{T}$ to pairs of non-crossing segments on P, in general a k-dimensional simplex of $\mathbb{T}$ corresponds to $k+1$ pairwise non-crossing segments on P and, finally, the facets of $\mathbb{T}$ correspond to triangulations of the point set P. The complex $\mathbb{T}$ is $(m-1)$-dimensional, where m is the number of edges in any triangulation of P, including the edges on the convex hull.

In what follows, let S be a set of d non-crossing fixed edges on P, contained in both given triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ that must stay fixed during flipping. Throughout, we assume that S does not contain any convex hull edges, so $m - d \geq 3$.

The set S corresponds to a $(d-1)$-dimensional simplex σ_S in the simplicial complex $\mathbb{T}$. The link $lk(\sigma_S)$ in $\mathbb{T}$ corresponds to a set of plane graphs H on P, with edge set $E(H)$, which are edge-disjoint from S and such that $H \cup S$ is plane:

$$lk(\sigma_S) = \{H : E(H) \cap E(S) = \emptyset \text{ and } E(H) \cup E(S) \text{ is non-crossing}\}.$$

We denote $\mathbb{T}_S := lk(\sigma_S)$ and call $\mathbb{T}_S$ the *simplicial complex of plane graphs on P constrained to the set S*. Note that $\mathbb{T}_S$ is a subcomplex of $\mathbb{T}$.

The plan is to prove that $\mathbb{T}_S$ is a shellable $(m-d-1)$-dimensional pseudomanifold with non-empty boundary, from which it will follow by Proposition 11 that $\mathbb{T}_S$ is a piecewise-linear ball. We then characterize the interior faces of $\mathbb{T}_S$, form the dual regular cell complex $\mathbb{X}_S$ of $\mathbb{T}_S$, prove the key topological properties of $\mathbb{X}_S$ in an analogue of Theorem 7 and, finally, we prove the Constrained Decomposition Theorem 19.

Lemma 20. *The simplicial complex $\mathbb{T}_S$ (a subcomplex of $\mathbb{T}$) is an $(m-d-1)$-dimensional pseudomanifold, i.e., $\mathbb{T}_S$ is pure and every $(m-d-2)$-dimensional face of $\mathbb{T}_S$ is contained in at most two $(m-d-1)$-faces.*

Proof. The highest-dimensional faces of $\mathbb{T}_S$ correspond to plane graphs on P with $m-d$ edges (triangulations on P constrained to S, with edges of S removed), so $\mathbb{T}_S$ is an $(m-d-1)$-dimensional simplicial complex.

We show that each face of $\mathbb{T}_S$ is contained in some $(m-d-1)$-dimensional facet. It is well known that any plane graph on point set P can be completed to a triangulation by adding edges. Hence, also any plane graph H corresponding to a face h of $\mathbb{T}_S$ can be completed to a plane graph H' having $m-d$ edges disjoint from S: starting with H, add edges of S, then complete to an arbitrary triangulation, then delete edges of S. The resulting graph H' has $m-d$ edges and corresponds to a facet h' of $\mathbb{T}_S$ containing the face h, hence, $\mathbb{T}_S$ is pure.

Finally, any $(m-d-2)$-dimensional face of $\mathbb{T}_S$ corresponds to a plane graph H with $m-d-1$ edges, disjoint from S. There are at most two ways in which to add an edge to H so that it stays disjoint from S, this is because there are at most two ways to complete the graph $H \cup S$ (that is a triangulation with a missing edge) to a triangulation. Hence, any $(m-d-2)$-dimensional face of $\mathbb{T}_S$ is contained in at most two $(m-d-1)$-faces and $\mathbb{T}_S$ is a pseudomanifold. $\square$

Lemma 21. *$\mathbb{T}_S$ is shellable.*

Proof. Recall that to prove shellability, we need to exhibit an order $F_1, F_2, \ldots, F_N$ of the $(m-d-1)$-dimensional facets of $\mathbb{T}_S$ such that for every $2 \leq j \leq N$ and every $i < j$, there exists some $k < j$ such that $F_i \cap F_j \subseteq F_k \cap F_j$ and $F_k \cap F_j$ is $(m-d-2)$-dimensional.

Complete facets $F_1, F_2, \ldots, F_N$ to triangulations $T_{F_1}, T_{F_2}, \ldots, T_{F_N}$ by adding the edges of S and with every triangulation T, associate the sorted vector of angles $\alpha(T) = (\alpha_1(T), \alpha_2(T), \ldots, \alpha_{3t}(T))$ where $\alpha_1(T) \leq \alpha_2(T) \leq \cdots \leq \alpha_{3t}(T)$ are the angles occuring in the triangulation T. Recall that we assume the point set P to be in general position. Let

$$T_{F_1}, T_{F_2}, \dots, T_{F_N} \tag{2.2}$$

be a total ordering of the triangulations sorting the corresponding angle vectors lexicographically from largest to smallest:

$$\alpha(T_{F_1}) >_{LEX} \alpha(T_{F_2}) >_{LEX} \cdots >_{LEX} \alpha(T_{F_N}).$$

Then the sequence contains exactly all triangulations constrained with respect to set S and, in particular, T_{F_1} is the constrained Delaunay triangulation with respect to S.

Given $F := F_i \cap F_j$, consider the subsequence $T_{F_{k_1}}, T_{F_{k_2}}, \dots$ of the sequence (2.2) consisting only of those triangulations that contain the edge set $F \cup S$. Then $T_{F_{k_1}}$ is the constrained Delaunay triangulation with respect to $F \cup S$, triangulations T_{F_i} and T_{F_j} both appear in the subsequence and, in particular, $T_{F_j} \neq T_{F_{k_1}}$ because T_{F_i} precedes it. Since every triangulation containing $F \cup S$ can be tranformed to the constrained Delaunay triangulation $T_{F_{k_1}}$ (see, for example, Dyn et al. [52] and a summary on constrained triangulations in Sections 2.2 - 2.3), there must exist an edge $e \in T_{F_j} \setminus T_{F_{k_1}}$ such that flipping e (Lawson flip) increases the angle vector. Then the triangulation resulting from flipping e in T_{F_j} is some T_{F_k} with $k < j$, satisfies $F \cup S \subseteq T_{F_k} \cap T_{F_j}$ and $T_{F_k} \cap T_{F_j}$ is $(m-2)$-dimensional since the triangulations differ by a single flip.

Then, removing the edges of S, we obtain $F = F_i \cap F_j \subseteq F_k \cap F_j$ and $F_k \cap F_j$ is $(m-d-2)$-dimensional, as desired. $\square$

Recall the definition of the boundary and interior faces of a piecewise-linear pseudomanifold $\mathbb{B}$ of dimension k: the *boundary* $\partial\mathbb{B}$ of $\mathbb{B}$ is the subcomplex of $\mathbb{B}$ consisting of all faces F for which there exists a $(k-1)$-dimensional face G of $\mathbb{B}$, with $F \subseteq G$, such that G is contained in a unique k-dimensional face of $\mathbb{B}$. A face F of $\mathbb{B}$ that does not lie in $\partial\mathbb{B}$ is called an *interior face*.

Lemma 22. *$\mathbb{T}_S$ has non-empty boundary.*

Proof. We need to exhibit an $(m-d-2)$-dimensional face that is contained in only one $(m-d-1)$-face. Since the edge set S does not contain any convex hull edges of P, any $(m-d-1)$-dimensional face h of $\mathbb{T}_S$ corresponds to a plane graph H containing all convex hull edges. Then the plane graph $H-e$, where e is a convex hull edge, corresponds to an $(m-d-2)$-dimensional face h' of $\mathbb{T}_S$ that is contained in a single $(m-d-1)$-face of $\mathbb{T}_S$. $\square$

To summarize, Lemmas 20, 21 and 22, together with Proposition 11, imply:

Theorem 23. *The simplicial complex $\mathbb{T}_S$ is a shellable $(m-d-1)$-dimensional pseudomanifold with non-empty boundary, and hence, it is a piecewise-linear ball.*

We next characterize the interior faces of $\mathbb{T}_S$, since these will correspond to the cells in the dual complex.

Proposition 24. *Let $\mathbb{T}_S$ be the simplicial complex of plane graphs on P constrained to the set S. A face of $\mathbb{T}_S$, i.e., a non-crossing set of edges F on P such that $F \cup S$ is plane and S and F are edge-disjoint, is an interior face of $\mathbb{T}_S$ if and only if the following conditions hold:*

(i) F contains all convex hull edges of P,

(ii) Every bounded region in the complement of the plane graph $(P, F \cup S)$ is convex.

Proof. (The proof is identical to the proof of Proposition 13, with the exception of considering a boundary face of the *constrained* complex $\mathbb{T}_S$ and considering plane graphs $F \cup S$ on P being completed to a triangulation, instead of just F.)

Similarly as in the proof for the original complex $\mathbb{T}$, also here we prove the negation of the statement: F is a boundary face of $\mathbb{T}_S$ if and only if F misses a convex hull edge or $F \cup S$ has an interior point p of P with no edge in a half-plane, where "having no edge in a half-plane" is defined as in the original proof.

If F is a boundary face of $\mathbb{T}_S$, then there is a triangulation T with $F \cup S \subseteq T$, and an edge $e \in T \setminus (F \cup S)$ such that e is non-flippable in T. If e is a convex hull edge, then F does not contain all convex hull edges. Otherwise e is a diagonal of a non-convex quadrilateral in T. Set p to be the reflex vertex of the non-convex quadrilateral and H to contain the other end of e but not the two other vertices of the quadrilateral. Then p has no edge in the half-plane H in $T \setminus e$ and hence, also in $F \cup S$.

For the other direction, first note that if F misses a convex hull edge then F is a boundary face in $\mathbb{T}_S$. For the other case, suppose that $F \cup S$ has a non-convex hull point p of P that has no edge in the half-plane H. Augment $F \cup S$ to a maximal set F' of non-crossing edges without using any edge from p into H. This will not yet be a triangulation. Now augment F' further to a triangulation T. Then $T \setminus F'$ contains some edge e incident to p, and e is not flippable in T otherwise we could have further augmented F'. Thus F is a boundary face in $\mathbb{T}_S$. $\square$

We next define the *flip complex constrained to S,* denoted by $\mathbb{X}_S$, as the dual complex of the piecewise-linear ball $\mathbb{T}_S$. This is done exactly as in the original proof, see Proposition 14: every interior k-dimensional face of $\mathbb{T}_S$ gives rise to a dual cell of dimension $(\dim \mathbb{T}_S) - k$ and the dual cells form a regular cell complex $\mathbb{X}_S$.

The following is a key result on topological properties of the constrained flip complex $\mathbb{X}_S$, analogous to Theorem 7. The elementary 4- and 5-cycles in the flip graph constrained to S are defined as in the original proof.

Theorem 25. *Let P be a set of n points in general position in the plane. There is a high-dimensional cell complex $\mathbb{X}_S = \mathbb{X}_S(P)$, which we call the* flip complex constrained to S*, such that:*

1. The 1-skeleton of $\mathbb{X}_S$ is the flip graph of P constrained to the set of edges S;

2. There is a one-to-one correspondence between the 2-cells of $\mathbb{X}_S$ and the elementary 4-cycles and elementary 5-cycles of the flip graph of P constrained to S;

3. $\mathbb{X}_S$ has the topology of (i.e., is homotopy equivalent to) a high-dimensional ball; therefore its fundamental group*, $\pi_1(\mathbb{X}_S)$, is trivial.*

Proof. (The proof is almost identical to the proof of Theorem 7 on page 75, only instead of considering the $(m-1)$-, $(m-2)$- and $(m-3)$-dimensional interior faces of complex $\mathbb{T}$ and the plane graphs with m, $m-1$ and $m-2$ edges, we consider the $(m-d-1)$-, $(m-d-2)$- and $(m-d-3)$-dimensional interior faces F of $\mathbb{T}_S$ and the corresponding plane graphs $F \cup S$ with m, $m-1$ and $m-2$ edges.)

By Theorem 23, $\mathbb{T}_S$ is a piecewise-linear ball and thus it has a trivial fundamental group. By Proposition 14, $\mathbb{X}_S = \mathbb{T}_S^*$ is a regular cell complex that is homotopy equivalent to the ball $\mathbb{T}_S$. Consequently (using the fact that homotopy equivalent spaces have isomorphic fundamental groups, see, for example, [72]), the fundamental group $\pi_1(\mathbb{X}_S)$ is trivial.

It remains to show the characterization of the vertices, edges and 2-cells of $\mathbb{X}_S$.

The vertices of $\mathbb{X}_S$ correspond (are dual) to the faces of $\mathbb{T}_S$ of the highest dimension $m-d-1$ and so are automatically interior faces of $\mathbb{T}_S$. The $(m-d-1)$-dimensional faces of $\mathbb{T}_S$ correspond to plane graphs with $m-d$ edges on P that together with edges in S form a triangulation. Hence, the vertices of $\mathbb{X}_S$ are in bijection with the triangulations of P constrained to S.

The edges of $\mathbb{X}_S$ correspond to interior $(m-d-2)$-dimensional faces F of $\mathbb{T}_S$, i.e., faces F such that the plane graph $F \cup S$ is contained in two triangulations of P that differ by a flip. Thus, the 1-skeleton of $\mathbb{X}_S$ is exactly the flip graph of P constrained to S.

Every 2-cell of $\mathbb{X}_S$ is the dual cell F^* of an interior face F of $\mathbb{T}_S$ of dimension $m-d-3$. Consider an arbitrary triangulation T containing $F \cup S$, i.e., $F \cup S$ is obtained from T by deleting two edges e, f. By Proposition 24, e and f are both flippable in T since they lie in a convex polygon in T.

If e and f are not incident to a common triangle in T, (or, equivalently, removing both e and f from T creates two internally disjoint convex quadrilaterals) then there exist four triangulations containing $F \cup S$ and these form an elementary 4-cycle in the constrained flip graph. It follows from the definition of $\mathbb{X}_S = \mathbb{T}_S^*$ that the 4-cycle is the boundary of the dual cell F^*.

Otherwise, e and f are incident to a common triangle in T. By Proposition 24 the union of the three triangles of T containing either e or f forms a convex polygon, necessarily a pentagon. There are five triangulations containing $F \cup S$ and these form an elementary 5-cycle in the constrained flip graph. It follows from the definition of $\mathbb{X}_S = \mathbb{T}_S^*$ that the 5-cycle is the boundary of the dual cell F^*.

Hence, every 2-cell of $\mathbb{X}_S$ corresponds to an elementary 4- or 5-cycle of the flip graph constrained to S.

Conversely, every elementary 4- and 5-cycle of the constrained flip graph gives rise to a 2-cell F^* of $\mathbb{X}_S$: more precisely, F^* corresponds to the intersection of the triangulations in the elementary cycle with deleted edges of S. □

The *elementary quadrilateral/pentagonal walks* in the flip graph constrained to S are defined as before: they are closed walks in the constrained flip graph of the form

wzw^{-1} where z is an elementary 4- or 5-cycle in the constrained flip graph and w is a walk from some triangulation T_0 to some triangulation on z. As before, the elementary quadrilateral walks do not permute labels and the elementary pentagonal walks perform elementary swaps of labels (Lemma 8). Spurs do not permute labels (Lemma 9). The proofs of Lemmas 8 and 9 carry over to the constrained scenario.

Finally, the Constrained Decomposition Theorem 19 follows from Theorem 25 by the same argument as the Decomposition Theorem 10 followed from Theorem 7, see the proof of Theorem 10 on page 69.

This concludes the proof of the Orbit Theorem 16 for Constrained Triangulations.

2.12 Shortest flip sequences in constrained versus unconstrained triangulations

We conclude the chapter by comparing the shortest flip sequences for constrained versus unconstrained triangulations. We have observed in previous sections that a flip sequence reconfiguring one triangulation into another exists even if certain set of edges cannot be flipped. For unlabelled triangulations this was guaranteed by the fact that the flip graph for constrained triangulations is connected, see Section 2.3. For labelled triangulations, existence of a flip sequence under certain conditions is guaranteed by the Orbit Theorem for constrained triangulations in Section 2.11.

In this section we show that the length of a flip sequence to reconfigure one triangulation of a point set into another may increase if we insist that some edges cannot be flipped. This is the case both in the labelled as well as unlabelled setting since the constrained edges, even though sitting in their correct position, can obstruct a more efficient flip sequence. An exceptional case are the unlabelled triangulations of a convex point set, in which the shortest flip sequences never flip edges that are already in the correct position, see below. This section provides a parallel in terms of triangulation flipping to an analogous problem of swapping ‘happy’ tokens on graphs that we study in Section 3.4 of Chapter 3 (although to define a truly analogous problem, one would have to consider constraining only such edges in a triangulation that play the role analogous to the role of leaf vertices in a graph).

Given a pair of unlabelled triangulations of a point set, we call an edge e *happy* if it appears in both triangulations. In the case of labelled triangulations, e is *happy* if it appears in both triangulations and has the same label.

By a result of Sleator et al. [129] fixing happy edges in unlabelled triangulations of a convex point set does not affect the flip distances.

Lemma 26 (Lemma 3b in [129]). *If T_1 and T_2 are triangulations of a convex point set that have an edge e in common, then a shortest flip sequence between T_1 and T_2 never flips e.*

In fact, Sleator et al. [129] proved that if a happy edge was flipped, the resulting flip sequence would be at least two flips longer than the shortest possible flip sequence.

On the other hand, the shortest flip sequences between triangulations of a general point set may need to flip happy edges. Lubiw and Pathak [101] give an example of a 'capped channel' which consists of two reflex vertex chains and an extra vertex on the left that sees all vertices inside the polygonal channel, see Figure 2.9 where the example consists of six vertices on each vertex chain.

In the figure, if we require that all the happy edges, including the edge ab and the reflex chain edges (in bold) stay fixed, then reconfiguring triangulation T_1 to T_2 takes at least 25 flips. This can be seen by an argument by Hurtado et al. [83], where each triangle in the interior of the channel gets assigned either '0' or '1' according to whether it contains two vertices of the lower or the upper chain. The labelled triangles are ordered left-to-right and a flip always swaps a neighbouring '0'-triangle with '1'-triangle or vice versa. Hence, reconfiguring triangulation T_1 into T_2 is equivalent to inverting the sequence 1111100000 into 0000011111 by transposing adjacent elements which takes at least 25 such adjacent transpositions.

If, however, we allow to flip the happy edge ab, reconfiguring triangulation T_1 into T_2 can be done in 20 flips by flipping into a canonical triangulation as was used by Lubiw and Pathak [101]. The flip sequence in Figure 2.10 shows 10 flips to transform T_1 into the canonical form where all edges inside the polygonal channel are incident to vertex c and another 10 flips to transform it into triangulation T_2.

The same result holds for labelled triangulations, i.e., the shortest flip sequences do not necessarily respect the labelled happy edges. The example in Figure 2.11 consists

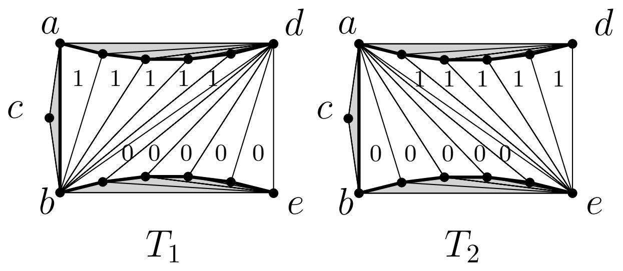


Figure 2.9: Two triangulations of a capped channel. Labelling the triangles inside the channel by ‘0’ or ‘1’ shows that at least 25 flips are necessary to reconfigure T_1 into T_2 if the happy edges remain fixed.

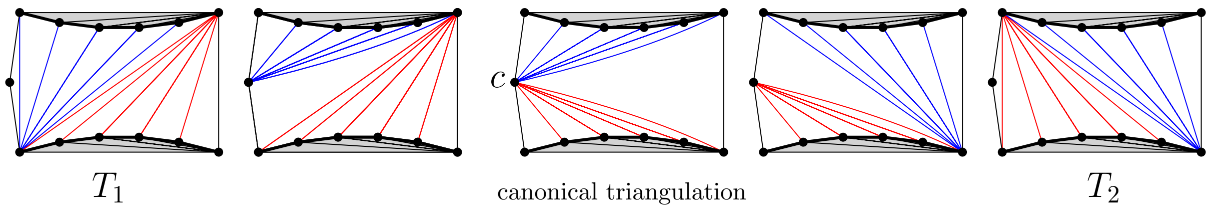


Figure 2.10: Reconfiguring triangulation T_1 into T_2 is possible in 20 flips if the flip sequence goes via the canonical triangulation.

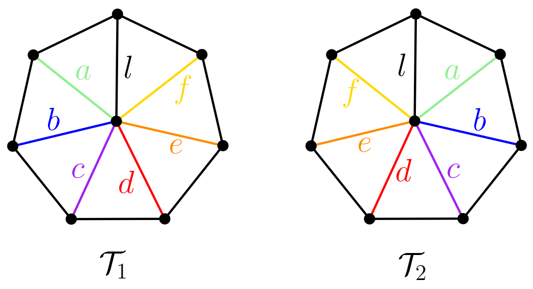


Figure 2.11: Two labelled triangulations of a regular 7-gon with a point in the middle for which the shortest reconfiguring flip sequence does not fix the happy label l. A shortest reconfiguration sequence consists of 37 flips. If the happy edge l must remain fixed, 55 flips are required to reconfigure $\mathcal{T}_1$ to $\mathcal{T}_2$.

of 7 points that are vertices of a regular 7-gon and a point in the middle. In both triangulations the middle point is connected to all other points, but, apart from one happy edge l, the edges are labelled in reversed order in the two triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$. When all edges can be flipped, it is pretty straightforward to find a reconfiguration sequence consisting of 37 flips. The idea is that the labels a and f can be swapped within the upper region of the 7-gon. If, however, the happy edge l must remain fixed, the labels a and f must instead travel around the 7-gon. Then the shortest flip sequence between $\mathcal{T}_1$ and $\mathcal{T}_2$ has 55 flips, as was confirmed by a program implementation. The program constructs the constrained labelled flip graph of the point set in which each triangulation contains the black happy edge labelled l (as in Figure 2.11), and checks with the Dijkstra algorithm that the shortest path between triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ in the flip graph has length 55.

Hence, we showed that the shortest flip sequences in the constrained setting, if they exist, may in general not be the shortest flip sequences between the given triangulations; and that this is the case in both the labelled and unlabelled setting. We will revisit this question in an analogous setting of swapping happy tokens on trees in Section 3.4 of Chapter 3.

3 Token Swapping on Trees

This chapter is about reconfiguring tokens placed on vertices of graphs, and, specifically, on trees. Throughout we assume that there is one token placed on each vertex and the tokens are labelled $1, \dots, n$, where n is the number of vertices. The reconfiguration step is to swap two tokens on adjacent vertices, hence the name *token swapping*.

We present new results related to computing shortest reconfiguration sequences for token swapping on trees: we disprove the Happy Leaf Conjecture, and discuss that computing the shortest reconfiguration sequences for *coloured weighted* token configurations is NP-hard on trees, while showing that it is solvable in polynomial time for paths and stars.

We start in Section 3.1 by providing basic definitions, and by describing the main reconfiguration questions in the context of token swapping. Section 3.2 gives a short survey of past results on token swapping on general graphs as well as on trees and other special classes of graphs.

Sections 3.3 – 3.7 cover the new results. We start with a counterexample to the Happy Leaf Conjecture in Section 3.4 and its generalization in Section 3.5 that demonstrates the importance of swapping the already-correctly-placed leaf tokens in order to obtain optimal reconfiguration sequences. As a step towards establishing whether the problem of shortest token swapping reconfiguration on trees lies in P or is NP-complete, we consider a generalization of the problem in Section 3.6 – the weighted coloured token swapping. Our paper [24] shows that computing the shortest reconfigurations in the generalized setting is NP-hard on trees, and we provide polynomial time algorithms for paths and stars in Section 3.7.

For a survey on other variants of token reconfigurations on graphs, such as, for example, token sliding or jumping, involving varying number of (possibly unlabelled) tokens or constrained token placements, see Section 1.5.

3.1 Introduction to token swapping and reconfiguration set-up

Suppose we wish to sort a list of numbers and the only allowable operation is to swap two adjacent elements of the list. It is well known that the number of swaps required is equal to the number of *inversions* in the list, i.e., the number of pairs that are out of order. Many other problems of sorting with a restricted set of operations have been studied, for example, pancake sorting, where the elementary operation is to flip a prefix of the list; finding the minimum number of pancake flips for a given list was recently proved NP-complete [36].

A much more general problem arises when we are given a set of generators of a permutation group, and asked to express a given permutation π in terms of those generators. Although there is a polynomial time algorithm to test if a permutation can be generated, finding a minimum length generating sequence was proved PSPACE-complete in 1985 [86].

This chapter is about a problem, known recently in the computer science community as *token swapping*, that is intermediate between sorting a list by swaps and general permutation generation. The input is a graph with n vertices $v_1, \ldots, v_n$. There are n tokens, labelled $1, 2, \ldots, n$, and one token is placed on each vertex. The goal is to "sort" the tokens, which means getting token i on vertex v_i, for all $i = 1, \ldots, n$. The only allowable operation is to *swap* the tokens at the endpoints of an edge, i.e., if $e = (v_i, v_j)$ is an edge of the graph and token k is at v_i and token l is at v_j, then we can move token k to v_j and token l to v_i. See Figure 3.1. The *token swapping problem* is to find the minimum number of swaps to sort the tokens. In the special case when the graph is a path, the token swapping problem is precisely the classic problem of sorting a list using adjacent swaps, see Knuth [94].

In the terminology of reconfiguration, the reconfiguration graph has as vertices all possible token-to-vertex assignments on the given graph. Two assignments are adjacent if they differ by a single swap. The token swapping problem asks to compute the

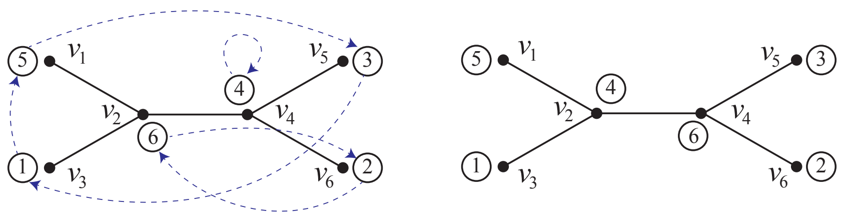


Figure 3.1: An example of the token swapping problem. Left: a tree of 6 vertices and an initial placement of tokens (in circles) on the vertices. Blue dashed arrows indicate where each token should go. Token 4 is home. The corresponding permutation is $(1\ 5\ 3)(2\ 6)(4)$. Right: the effect of swapping tokens 4 and 6. Now token 6 is closer to its destination but token 4 is further from its destination. One swap sequence that sorts the tokens to their destinations is (4 6), (6 2), (2 4), (3 4), (3 2), (3 1), (1 5), (5 2), (5 4). This sequence has 9 swaps, but there is a swap sequence of length 7.

shortest path between two vertices in the reconfiguration graph (where, without loss of generality, one token-vertex assignment is assumed to be the identity permutation).

The reconfiguration graph for token swapping on a graph G is also known as the *Cayley graph of transposition graph* G. In general, Cayley graphs are defined for any group and its generating set. Recall that, given a group $(F, *)$, a subset S is a *generator* of F, if every element of F can be expressed as the product of finitely many elements of S and their inverses. Given a group F and a generator S of F, the *Cayley graph* $\Gamma(F, S)$ has the elements of F as vertices and any two vertices v, w are adjacent, if there exists an element $s \in S$ such that $v * s = w$ (by default, the edges are undirected). In our context, we are interested in the Cayley graph of the symmetric group S_n that consists of all permutations of the n element set $\{1, \ldots, n\}$. The generating set S_G is determined by a graph $G = (V, E)$ on n vertices, called a *transposition graph*: S_G is defined as the set of all transpositions corresponding to edges of E. It can be shown that S_G is a minimal generating set for the group S_n if and only if the transposition graph is a tree. The Token swapping problem corresponds to finding the shortest path in the Cayley graph $\Gamma(S_n, S_G)$ from a given permutation π to the identity permutation, where the permutation is $\pi(i) = j$ if token j is initially at vertex v_i. Note that this shortest path corresponds to the minimum length generating sequence of π by elements of S_G. The worst case minimum number of swaps between two token-vertex assignments corresponds to the diameter of the Cayley graph.

The following is a short overview of typical reconfiguration problems. Compared to triangulation reconfiguration in 2.1, some of the problems become easy or even trivial in the context of token swapping. On the other hand, as demonstrated in Section 3.2, the shortest paths and the diameter questions are a topic of active research in multiple disciplines.

Size of reconfiguration graph, connectivity and finding some swap sequence between two token assignments. Token swapping on a graph G with n vertices corresponds to a reconfiguration graph of size $n!$ which is connected if and only if G is connected. If a reconfiguration sequence exists, it takes at most $O(n^2)$ swaps, see Section 3.2.

Diameter of the reconfiguration graph. Bounding the worst-case number of swaps for a given graph and, in particular, a tree, over all possible token placements, is of interest in the community of sorting networks. Akers and Krishnamurthy [9] gave the first bounds for diameters of Cayley graphs of transposition trees in 1989. The bounds and their computation time have subsequently been improved by Ganeson [65], Chitturi [45] and Kraft [96].

Computing the distance and shortest paths in the reconfiguration graph. The Token swapping problem on graphs was proved NP-complete [13], and even APX-hard [106], in 2016, and further hardness results have appeared since then [28]. There are polynomial time algorithms for paths, cliques [40], cycles [86], and stars [9; 120; 116], and some other special cases.

Token swapping on a tree is not known to be in P or NP-complete. Several papers have given polynomial time 2-approximation algorithms for trees [9; 141; 148; 106].

The token swapping problem has been generalized in several ways. In *weighted token swapping* each token i has a positive weight $w(i)$ and the cost of swapping token k and token l is $w(k) + w(l)$. The goal is to sort the tokens while minimizing the sum of the costs of the swaps. In *coloured token swapping* [68; 149] the tokens have colours, and we are given an initial and final assignment of coloured tokens to the vertices. Tokens of the same colour are indistinguishable. The goal is to move from the initial to the final token arrangement using the fewest swaps. The original problem is the case where each token has a distinct colour. Coloured token swapping on graphs is NP-hard for 3 colours [149] but solvable in polynomial time for 2 colours. In *weighted coloured token swapping* we have coloured tokens and each colour has a weight. Such a weighted colored version has been studied for string rearrangements under various cost models, which allow swapping non-adjacent elements [12].

Token swapping on general graphs is an active topic that has been studied by different research communities in mathematics, computer science, and engineering, often unaware of each others' work. Broadly speaking, mathematicians aim to understand properties of Cayley graphs, computer scientists study algorithmic aspects of token swapping. In particular, the network community is interested in using Cayley graphs of transposition trees as interconnection networks, and the robotics community can apply some of the algorithms to robot motion planning. We survey all the results about token swapping that we know of in Section 3.2. Section 1.5 covers some related results about more general token reconfigurations.

Our results contained in Sections 3.3–3.7 are about token swapping on a tree and, in particular, our emphasis is on computing the number of swaps, and the actual swap sequence, needed for a given placement of tokens on a graph. As explained above, this problem is equivalent to finding a shortest path in the Cayley graph of a transposition tree and is also known as 'sorting with a transposition tree'.

3.2 Survey of token swapping results

We start by describing results related to the diameter of Cayley graphs of transposition trees. Then we turn to the main topic of our interest – the token swapping problem – and describe the relevant results for general graphs, trees, paths and stars. Finally, we discuss the work on 'happy leaves' and coloured/weighted token swapping that we extend with our results in the following sections.

Transposition trees and interconnection networks

The sorting network community's interest in token swapping on trees ("sorting with a transposition tree") stems from the use of the corresponding Cayley graphs as interconnection networks. Specifically, the Cayley graph of a star (a tree with one non-leaf) is a good alternative to a hypercube. Akers and Krishnamurthy [9] first introduced this idea in 1989, and their paper has been cited more than 1400 times according to Google scholar.

Cayley graphs of transposition trees have the following desirable properties: they are large graphs ($n!$ vertices) that are vertex symmetric, with small degree ($n-1$), large connectivity (the same as the degree), and small diameter. In particular, the diameter is $\frac{3}{2}n + O(1)$ when the tree is a star. The commonly used hypercube has 2^n vertices and diameter n, so the diameter is logarithmic in the size. By contrast, the Cayley graph of a star has sublogarithmic diameter.

Akers and Krishnamurthy proved a bound on the diameter of the Cayley graph of a transposition tree, specifically, the maximum over all permutations of the bound $D - (n - c)$ which is defined in the section on the happy swap algorithm below. This bound cannot be computed efficiently since it involves the maximum over $n!$ permutations. Vaughan [139] also gave upper and lower bounds on the diameter of the Cayley graph, though neither easy to state nor to prove.

Follow-up papers by Ganesan [65], Chitturi [45] and Kraft [96] have lowered the diameter bound and/or the time required to compute the bound. To give a flavour of the results, we mention a polynomial-time computable upper bound, γ, due to Chitturi [45] that is defined recursively as follows: if the tree is a star, use the known diameter bound; otherwise choose a vertex v that maximizes the sum of the distances to the other vertices, increase γ by the maximum distance from v to another vertex and recurse on the smaller tree formed by removing the leaf v.

Token swapping on graphs

Token swapping on a connected graph of n vertices takes at most $O(n^2)$ swaps—take a rooted spanning tree and, for vertices in leaf-first order, successively *home* the token that goes to that vertex, where *homing* a token means swapping it along the unique path to its final location. This bound is tight for a path with tokens in reverse order. The token swapping problem on graphs (to compute the minimum number of swaps between two given labellings of the graph) is NP-complete, and in fact, APX-complete, as proved by Miltzow et al. [106]. They complemented these hardness results with a polynomial-time 4-approximation algorithm, and an exact exponential time algorithm that is best possible assuming ETH. These results extend to coloured token swapping. Bonnet et al. [28] showed that token swapping is W[1]-hard parameterized by number of swaps, but fixed parameter tractable for nowhere dense graphs. This result extends to coloured token swapping and even to a further generalization called “subset token swapping”.

There are many special classes of graphs on which token swapping can be solved via exact polynomial time algorithms. These include (in historical order): cliques [40], paths [94], cycles [86], stars [9; 120; 116], brooms [140; 92], complete bipartite graphs [148], and complete split graphs [152]. See the survey by Kim [93].

Token swapping on trees

The big open question is whether the token swapping problem on a tree is in P or is NP-complete. Various efficient but non-optimal algorithms for token swapping on a tree have been presented in the literature. Most of them are 2-approximations—i.e., they use at most twice the optimum number of swaps—although this was not always noted. Several of the algorithms are expressed in terms of the paths that tokens should take. For any token i, there is a unique path $p(i)$ from its initial vertex to its final vertex v_i. Let $d(i)$ be the length (the number of edges) of the path $p(i)$, and let $D = \sum_i d(i)$.

Happy swap algorithm. The earliest algorithm we are aware of is due to Akers and Krishnamurthy in 1989 [9]. Their algorithm involves two operations that we will call a “happy swap” and a “shove.” Let (u, v) be an edge with token i on u and token j on v. A *happy swap* exchanges i and j if $p(i)$ includes v and $p(j)$ includes u, i.e., the two tokens want to travel in opposite directions across the edge e as the first steps in their paths. A *shove* exchanges i and j if $p(i)$ includes v and j is home. Akers and Krishnamurthy show that: (1) one of these operations can always be applied; and (2) both operations decrease $M = D - (n - c)$ where n is the number of vertices and c is the number of cycles in the permutation π defined by $\pi(i) = j$ if token j is initially at v_i. Note that if $\pi(i) = i$ (i.e., i is home) this forms a trivial cycle which counts in c.

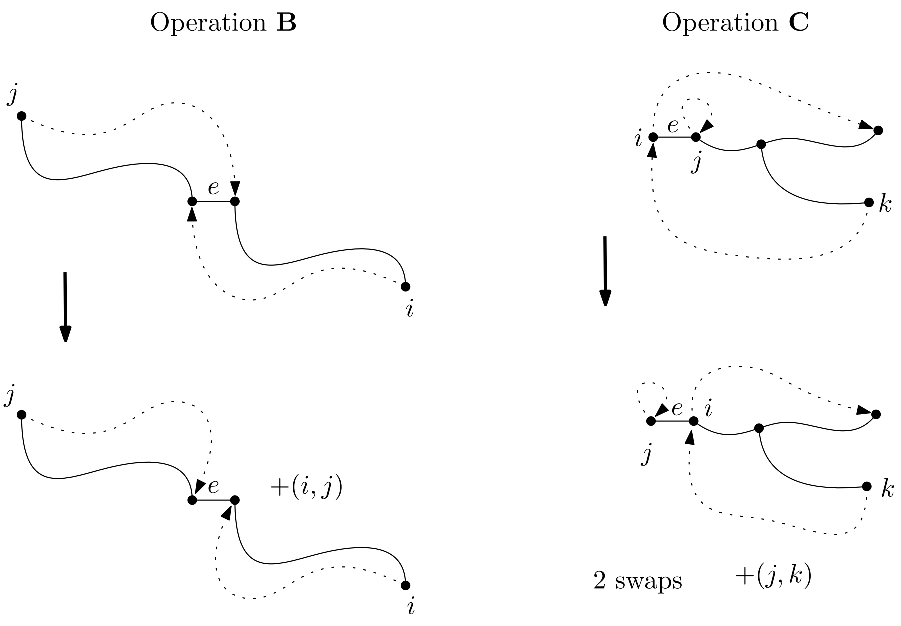


Figure 3.2: Illustration of Operation $\mathbf{B}$ and $\mathbf{C}$. The solid lines indicate paths in the tree and the dotted lines indicate the target vertex of the corresponding token.

Both aspects (1) and (2) of the proof are fairly straightforward. For (2) they prove that a shove does not change D but decreases c, whereas a happy swap decreases D by 2 and changes c by at most 1. Their proof implies that M is an upper bound on the minimum number of swaps. They do not claim that M is at most twice the minimum, but this follows from the easy observation that $M \leq D$ and $D/2$ is a lower bound on the minimum number of swaps, since a single swap decreases D by at most 2.

Miltzow et al. [106] gave a 4-approximation algorithm for [coloured] token swapping on general graphs. In case the graph is a tree, their algorithm is the same as the one of Akers and Krishnamurthy and they prove that it is a 2-approximation.

Vaughan's algorithm. Independently of the work by Akers and Krishnamuthy, Vaughan [141] in 1995 gave an algorithm for token swapping on a tree that uses a number of swaps between $D/2$ and D (in her notation D is called "PL"). Her algorithm involves three operations: **A**, a happy swap; **B**, a version of a happy swap that alters the final token assignment; and **C**, a variant of a shove. Her operations construct the swap sequence by adding swaps at the beginning and the end of the sequence, whereas the other algorithms construct the sequence from the start only.

Operation **B** applies when there is an edge $e = (u, v)$ and tokens i and j such that the destination of i is u and the destination of j is v and $p(i)$ includes v and $p(j)$

includes u, i.e., the two tokens want to travel in opposite directions across the edge e as the last steps in their paths. The operation exchanges the final destinations of i and j, computes a swap sequence for this subproblem, and then adds the swap of i and j at the end of the sequence.

Operation **C** applies in the following situation. Suppose there is an edge $e = (u, v)$ with token i on u and token j on v, where $p(i)$ includes v and token j is home. Suppose furthermore that there is a token k whose destination is u and whose path $p(k)$ includes v. (Note that this is a more restrictive condition than for a shove.) The operation exchanges tokens i and j and exchanges the final destinations of j and k. Recursively solve this subproblem. The swap sequence consists of the swap of i and j, followed by the sequence computed for the subproblem, followed by the swap of j and k.

Vaughan proves that if operations $\mathbf{A}$ and $\mathbf{B}$ do not apply, then operation $\mathbf{C}$ does, and she proves that each operation decreases the sum of the distances by 2.

Cycle algorithm. The first explicit description of a 2-approximation algorithm for token swapping on trees was given by Yamanaka et al. [148], who gave an algorithm that sorts the cycles of the permutation one-by-one. Consider a cycle of tokens $(t_1 t_2 \cdots t_q)$ in the permutation π. For $i = 1, \ldots, q-1$ their algorithm swaps token t_i along the path from its current vertex to the vertex currently containing token t_{i+1}—but stops one short of the destination. Finally, token t_q is swapped from its current vertex to its (original) destination.

We now outline their proof of correctness and the bound on the number of swaps. Suppose that token t_1 is currently at vertex x and that the first edge it wishes to travel along is $e = (x, y)$. Let j be the minimum index, $2 \leq j \leq q$ such that t_j wishes to travel in the opposite direction along e (and observe that j exists). Then the cycle is equivalent to $(t_1 \cdots t_j)$ followed by $(t_j \cdots t_q)$, where the second cycle is empty if $j = q$. Also, the algorithm performs the same swaps on these two cycles as on the original. Thus it suffices to prove that their algorithm correctly solves the cycle $(t_1 \cdots t_j)$. This cycle has the special feature that no tokens besides t_1 and t_j wish to traverse edge e. Yamamoto et al. prove that their algorithm “almost” achieves the property that just before step i (the step in which t_i moves) tokens $t_1, \ldots, t_{i-1}$ are at their final destinations and all other tokens, including the non-cycle tokens, are at their initial positions. “Almost” means that there is the following exception: there is one path from x along which the tokens are shifted by 1. Let z be the vertex containing t_i, and let z' be the next vertex on the path from z to x. All the tokens on the path from z' to x are one vertex away from their desired positions–they should all be one vertex closer to z. With this exception, the property is obvious for $i = 1$ and $i = 2$ and can be proved by induction, which implies that the algorithm is correct.

Because tokens are only "off-by-one" it can be argued that the number of swaps performed in step i of the algorithm is bounded by the original distance from t_i to its destination. This implies that the total number of swaps is at most the sum of the distances of labels in the cycle, which gives the factor 2 approximation.

Comparisons. A leaf in a tree that already has the correct token is called a *happy leaf*. None of the algorithms will swap a token at a happy leaf, so, as our example in Section 3.4 demonstrates, there is an instance where the algorithms are not optimal. The three algorithms differ in how far they allow a token i to stray from its path $p(i)$. In the happy swap algorithm no token leaves the set of vertices consisting of its path together with the vertices at distance 1 from its destination. In the cycle algorithm, no token leaves its path together with vertices at distance 1 from its origin and destination. In Vaughan's algorithm, a token may go further away from its path.

Token swapping on paths

Token swapping on a path is the classic problem of sorting a list by transposing adjacent pairs. See Knuth [94, Section 5.2.2]. The minimum number of swaps is the number of inversions in the list. Curiously, a swap that decreases the number of inversions need not be a happy swap or a shove (as described above) and, on the other hand, there does not seem to be any measure analogous to the number of inversions that applies to trees more generally, or even to stars.

The diameter of the Cayley graph for token swapping on a path is $\Theta(n^2)$. Researchers have also studied the number of permutations with a given number of inversions [94], and the relationship between the number of inversions and the number of cycles [53; 19].

Token swapping on stars

A *star* is a tree with one non-leaf vertex, called the *center vertex*. We will need the following known result about token swapping on a star, which expresses the number of swaps as a function of the number of cycles in the permutation π. The formula is often written with a delta term whose value depends on whether the center vertex is happy or not, but we will express it more compactly.

Lemma 27 ([9; 120; 116]). *The optimum number of swaps to sort an initial placement of tokens on a star is $n_U + \ell$, where n_U is the number of unhappy leaves and ℓ is the number of cycles in the permutation that have length at least 2 and do not involve the center vertex.*

Proof sketch. Consider a cycle C of length at least 2 in the permutation of tokens and consider the corresponding vertices of the star. If the center vertex is not in C then the number of swaps to sort C is its number of leaves plus one. If the center vertex is in C then the number of swaps is the number of leaves in C. Because the cycles are independent, we can sum over all non-trivial cycles, which yields the stated formula. $\square$

It follows that the diameter of the Cayley graph for a star is $\frac{3}{2}n + O(1)$, which arises when all cycles have length 2. Further properties of Cayley graphs of stars were explored by Qiu et al. [122]. Portier and Vaughan [120] analyzed the number of vertices of the Cayley graph at each distance from the distinguished “sorted” source vertex (see also [143]). Pak [116] gave a formula for the number of shortest paths between two vertices of the Cayley graph.

Happy leaves

Vaughan [139] conjectured that a happy leaf in a tree need not be swapped in an optimal swap sequence. In fact she made a stronger conjecture [139, Conjecture 1] that if a tree has an edge (a, b) such that no token wishes to cross (a, b) (i.e., no path from a token to its destination includes edge (a, b)) then there is an optimal swap sequence in which no token swaps across (a, b). The Happy Leaf Conjecture is the special case where b is a leaf.

Smith [131, Theorem 9] claimed something stronger than the Happy Leaf Conjecture: that no optimal swap sequence would ever swap a happy leaf. But later he found an error in the proof [132], and gave an example of a small tree where there is an optimal swap sequence that performs a swap on a happy leaf. In his example, there is also an optimal swap sequence that does not swap the happy leaf so he did not disprove the Happy Leaf Conjecture.

Coloured token swapping

Many natural reconfiguration problems involve "coloured" elements, where two elements of the same colour are indistinguishable. Token swapping for coloured tokens was considered by Yamanaka et al. [150] (journal version [149]). They proved that the coloured token swapping problem is NP-complete for $c \geq 3$ colours even for planar bipartite graphs of maximum degree 3, but for $c = 2$ the problem is solvable in polynomial time, and in linear time for trees. On complete graphs, coloured token swapping is NP-complete [28] but fixed parameter tractable in the number of colours [149]. The complexity of coloured token swapping on trees is open.

3.3 Our results

Most of the results in this chapter appear in the paper by Biniaz et al. [24]. Some parts of the paper in which the author was not significantly involved but are closely related (such as the NP-hardness proof for weighted coloured token swapping on trees) are only briefly described in this thesis, but whenever this is the case, we point the reader to the full version of the paper. Other parts of the paper, such as on factors of approximation algorithms or an algorithm for a 'broom' graph have been omitted altogether.

One feature of all the algorithms for token swapping on trees—both the poly-time algorithms for special cases and the approximation algorithms for the general case—is that they never swap a happy leaf. As mentioned above, in 1991 Vaughan [139] conjectured that an optimal swap sequence never needs to swap a token at a happy leaf. We give a 10-vertex counterexample to this Happy Leaf Conjecture in Section 3.4

and a generalized example in Section 3.5. In the paper [24] we furthermore show that any algorithm that fixes the happy leaves has approximation factor at least $4/3$, and we show that the two best-known 2-approximation algorithms (the happy swap algorithm and the cycle algorithm) have approximation factor exactly 2. These results provide new insight that the difficult aspect of token swapping on trees is knowing when and how to swap happy leaves.

Next, we explore whether this difficult aspect can be used to prove NP-hardness. In Section 3.6 we outline our result from [24] that the generalized version of *weighted coloured* token swapping is NP-hard for trees. Furthermore, we show in Section 3.7 that this generalized version remains poly-time on paths and stars, which gives further evidence that trees really are harder than paths and stars.

Finally, in an attempt to expand the set of "easy" cases, we devised a polynomial time algorithm for token swapping on a *broom*—a star with an attached path—only to discover that this had been done by Vaughan [140] in 1999, and by Kawahara et al. [91] in 2016. Our simpler proof is in [24].

Preliminaries. We say that a token is *home* if it is at its destination. In a tree, *homing* a token means swapping it along the (unique) path from its current position to its destination.

We defined the token swapping problem as: move token i from its initial vertex to vertex v_i, with associated permutation $\pi(i) = j$ if token j is initially at v_i. An alternative formulation is in terms of an initial and final token assignment. Suppose s is an initial assignment of tokens to vertices, and f is a final assignment of tokens to vertices. The goal then is to move each token i from its initial vertex $s(i)$ to its final vertex $f(i)$. The associated permutation is $\pi(i) = s^{-1}(f(i))$. (Our first formulation just eases notation by assuming that $f(i) = v_i$.)

A solution to a token swapping problem is a sequence of swaps, $\sigma_1, \sigma_2, \ldots, \sigma_k$. Our convention is that, starting with the initial token assignment, we perform the swaps starting with σ_1 and ending with σ_k to get the tokens to their final positions. Equivalently, performing the transpositions starting with σ_k and ending with σ_1 generates the associated permutation.

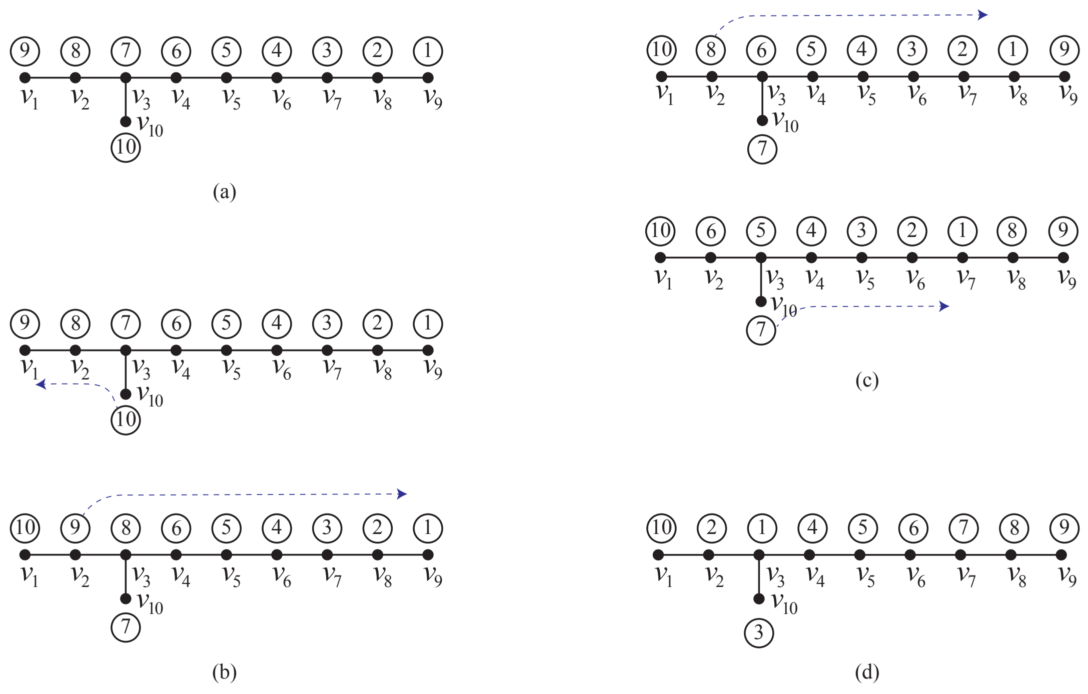


Figure 3.3: A counterexample to the Happy Leaf Conjecture where an optimum swap sequence involves moving the happy token 10. (a) The initial tokens (in circles). (b) Three swaps move token 10 to v_1. (Dashed arrows show the upcoming moves.) (c) The result of homing tokens 9 and 8. (d) The result of homing tokens 9 through 4. Four additional swaps will sort the tokens.

3.4 Counterexample to the Happy Leaf Conjecture

In this section we disprove the Happy Leaf Conjecture by giving a tree with initial and final token placements such that any optimal swap sequence must swap a token on a happy leaf (recall that a happy leaf is one that has the correct token). Our counterexample has $n = 10$ vertices and is shown in Figure 3.3(a). This is the smallest possible counterexample—we have verified by computer search that all trees on less than 10 vertices satisfy the Happy Leaf Conjecture. Our counterexample can easily be generalized to larger n.

Our tree consists of a path $v_1, \ldots, v_9$ and one extra leaf v_{10} joined by an edge to vertex v_3. The initial token placement has token 10 at v_{10} (so v_{10} is a happy leaf) and tokens $9, 8, \ldots, 1$ in that order along the path $v_1, v_2, \ldots, v_9$.

If token 10 does not leave vertex v_{10} (i.e., we fix the happy leaf), then we must reverse the order of the tokens on a path of length 9, which takes $\binom{9}{2} = 36$ swaps. However, as we now show, there is a swap sequence of length 34.

Initially, we perform 3 swaps to move token 10 to v_1 giving the configuration shown in Figure 3.3(b). Next, we ignore leaf v_1 and perform the sequence of swaps that homes tokens $9, 8, \ldots, 4$. The result of homing tokens 9 and 8 is shown in Figure 3.3(c), and the result of homing all of them is shown in Figure 3.3(d). It is easy to verify that this takes 7 swaps for token 9, 6 for token 8, $\ldots$, 2 for token 4, which adds up to $7 + 6 + \cdots + 2 = 27$ swaps.

Finally, we perform the following swaps to complete the sort: home token 10 in 3 swaps, then home token 1 in 1 swap. In total, this uses $3 + 27 + 4 = 34$ swaps.

The idea of why this saves swaps is as follows. To reverse the order of edges on a path, every token must swap with every other token. By contrast, the above approach saves swaps whenever two tokens occupy vertices v_2 and v_{10}. For example, tokens 8 and 7 never swap with each other, nor do 7 and 6, etc. We need $n \geq 10$ so that this saving exceeds the cost of the initial set-up and final clean-up.

3.5 Generalized counterexample

The swapping idea behind the counterexample from Section 3.4 that saves swaps is depicted again in Figure 3.4. Previously, each yellow and blue rectangle in the figure represented one token. Each rectangle can, however, also represent a *block* of tokens. The principle stays the same: always home the largest unhomed token. This is equivalent to swapping the entire block to the place along the path where it belongs and then sorting the tokens within the block. An actual swap sequence will additionally need to perform some extra swaps to initially set-up the tokens and, to clean-up at the end. Such a general full swap sequence is sketched in Figure 3.5.

In detail, the graph we consider is a path with an extra shorter path attached. Removal of the single degree-3 vertex splits the graph into three parts that we call a *branch* (down), *a short path* (left), and *a long path* (right). The blocks of tokens are chosen to have 3 different lengths. The blue blocks have length d, same as the branch. We assume that the short path has length $l + d$, where l is the length of the yellow blocks (so that one blue block plus one yellow block can exactly fit on the short path). There are two orange token blocks of length $l + 1$, originally placed as the very

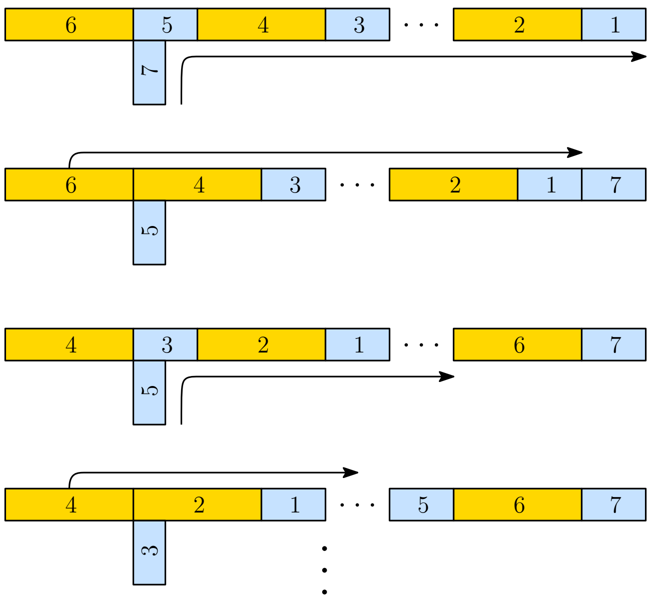


Figure 3.4: The swap-saving idea can also be used with blocks of tokens.

first and last block on the path, and needed for the initial setting-up swaps and final cleaning-up to work out. We assume that the long path has length $k(l + d) + l + 1$ for some $k > 0$, so that it fits exactly k repetitions of yellow-blue blocks and one orange block at the end.

As before, our aim is to reverse the tokens along the horizontal path, and we assume that the branch tokens start happy. See Figure 3.5. After the initial preparatory swaps we use the same swap-saving idea as in the original 10-vertex counterexample to reverse the blocks and close by a couple of auxiliary swaps.

The total number of swaps can be determined with a (bit lengthy but) straightforward computation. It turns out that the number of swaps saved, as compared to the standard swapping that fixes the happy tokens on the branch, is equal to $2kdl-d(d+1)$. Qualitatively, this makes sense: as seen in Figure 3.5, the blue blocks do not cross over any of the two neighboring yellow blocks. Since a crossing of a blue block with an yellow block costs dl swaps in total, the total number of swaps saved should be roughly $2kdl$, minus an overhead for preparation and clean up.

Such savings can be significant. In particular, if $k = 1$ and $d = l = n$ then the standard method uses $\binom{5n+1}{2} \sim 12.5n^2$ swaps and the improved method uses $2n^2 - n(n + 1) \sim n^2$ fewer swaps, which is a constant-factor improvement. In the most favourable case that we have found ($k = 2$, $d = 4n$, $l = 5n$) our method saves approximately $1/8$ of the swaps.

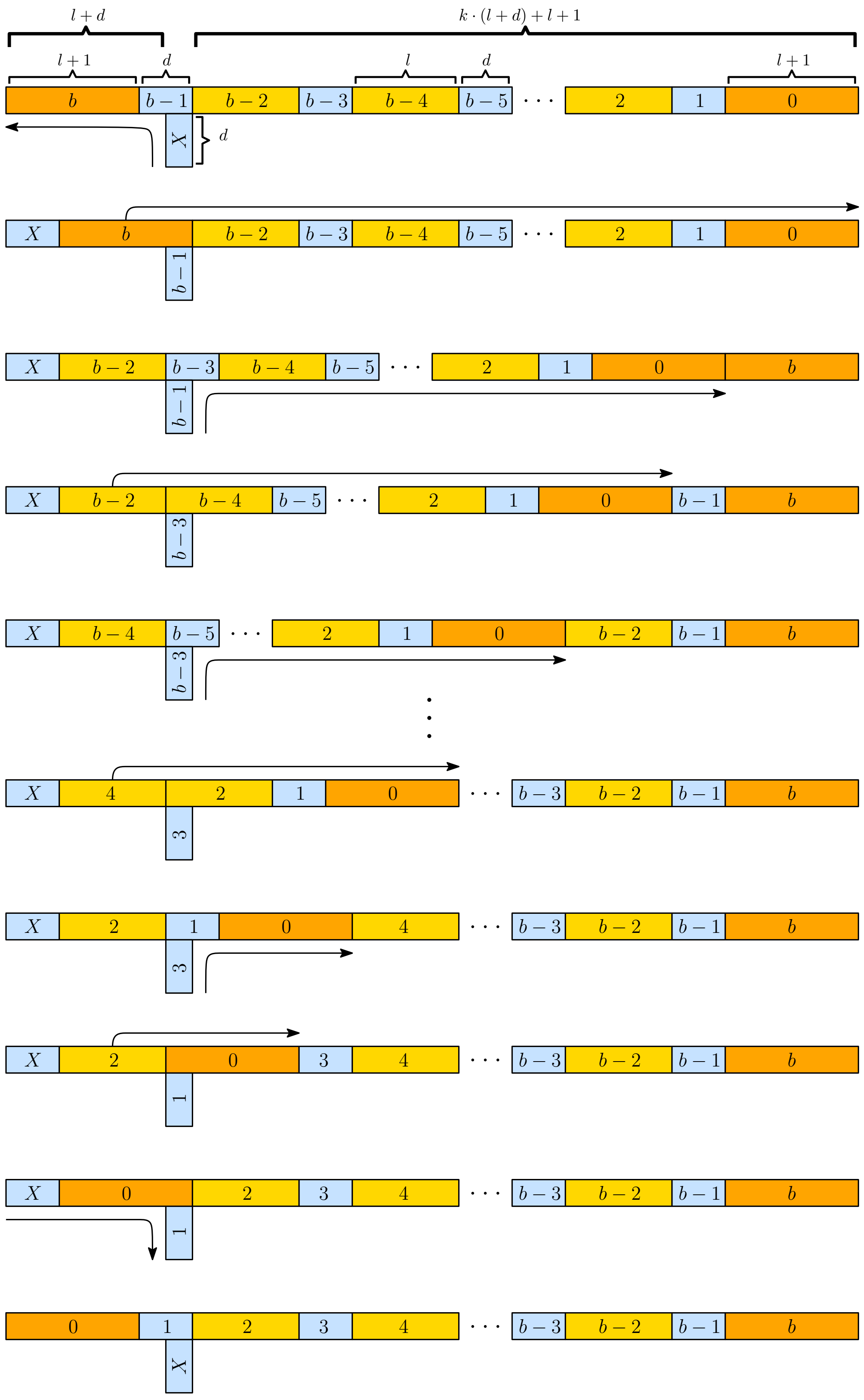


Figure 3.5: A generalized counterexample: X is the block of initially happy tokens. It is being stored at the beginning of the path, while the swap-saving idea from Figure 3.4 is used to reverse the yellow and blue tokens along the path. Eventually X is returned to its original place. Tokens within each block (apart from X) must be reversed in a standard way.

3.6 Weighted coloured token swapping is NP-hard

Recall that in the *coloured* token swapping problem, we have coloured tokens, one on each vertex of the graph, and each vertex has a colour. The goal is to perform a minimum number of swaps to get each token to a vertex that matches its colour. We assume that the number of tokens of each colour is equal to the number of vertices of that colour. The standard token swapping problem is the special case when all colours are distinct.

For general graphs the coloured token swapping problem can be solved in polynomial time for 2 colours, but becomes NP-complete for $k \geq 3$ colours [149]. See Section 3.2 for further background.

In the *weighted* coloured token swapping problem, each colour c has a weight $w(c)$, and the cost of swapping tokens of colours c and c' is $w(c)+w(c')$. The goal is to reach the target configuration with minimum total cost.

In [24] we proved that weighted coloured token swapping is NP-hard on trees. The idea is as follows: we reduce from Vertex Cover by constructing a long path with some green tokens initially at the right end of the path. The final configuration has green tokens at the left end of the path. We can save the cost of moving all those green tokens the whole length of the path by dislodging some happy green tokens from a subtree that dangles off the path; we construct this dangling subtree from the vertex cover instance in such a way that there is a cost savings in the token swapping problem if and only if there is a small vertex cover, see Figure 3.6.

3.7 Weighted coloured token swapping on paths and stars

In this section we give polynomial time algorithms for weighted coloured token swapping on paths and stars. Recall from the previous section our convention that we have coloured tokens and coloured vertices, with one token at each vertex, and with the number of tokens of each colour equal to the number of vertices of that colour. The goal is to perform swaps to get each token to a vertex that matches its colour. Each colour c has a weight $w(c)$ and the cost (or weight) of performing a swap on two tokens of colour c and c' is $w(c) + w(c')$. The objective is to minimize the total cost (weight) of

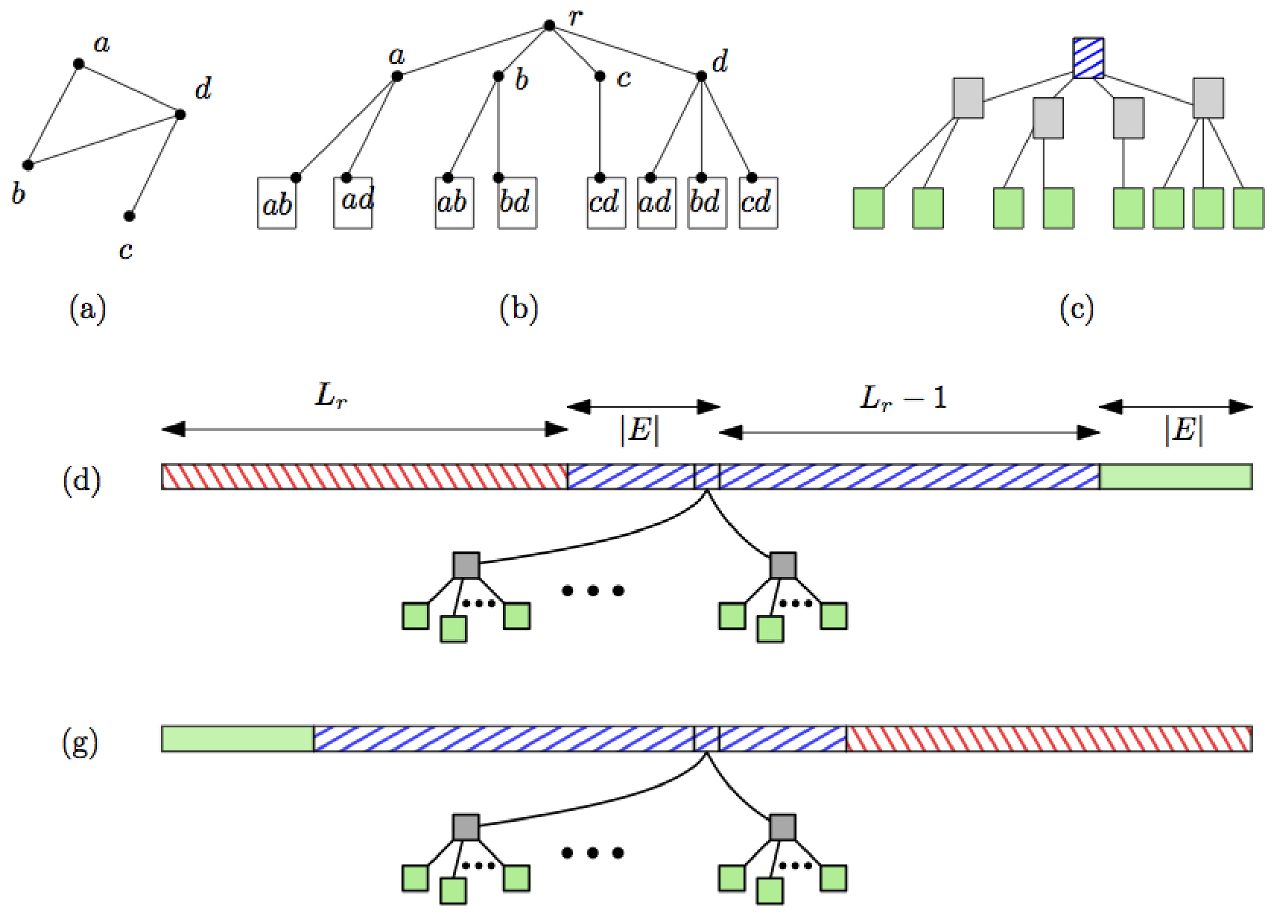


Figure 3.6: Illustration for the NP-hardness proof; (a) input graph G for Vertex Cover; (b) tree T corresponding to G to hang off a long path; (c) initial configuration of coloured tokens on T; (d) tree T' (obtained by attaching T to a long path) with initial configuration of coloured tokens; (g) T' with final configuration of coloured tokens. Note that the green tokens are, in fact, a collection of $|E(G)|$ different colours.

the swaps. Note that standard token swapping is the special case where all the colours are distinct and all the weights are $\frac{1}{2}$, since each swap moves two tokens.

The main issue in [weighted] coloured token swapping is to decide which token should go to which vertex. After fixing such a "token-vertex assignment" the problem becomes [weighted] token swapping without colours. In some situations—including for paths and stars—it turns out that the optimum token-vertex assignment does not depend on the weights. In these situations we can combine an algorithm for coloured token swapping and an algorithm for weighted token swapping to obtain an algorithm for weighted coloured token swapping.

Such a separation of colours and weights does not hold for trees in general, as the NP-hardness proof in the previous section shows. However, when the number of colours is 2, the weights and colours do separate—we should never swap two tokens of the same colour, and therefore every swap costs $w(c_1) + w(c_2)$ where c_1 and c_2 are the two colours. This means that, for 2 colours, weighted coloured token swapping is no harder than coloured token swapping. Yamanaka et al. [149] gave a polynomial-time algorithm for 2-coloured token swapping on general graphs. Thus, weighted 2-coloured token swapping can also be solved in polynomial time.

Our main result in this section is an algorithm for weighted coloured token swapping on stars. Before that, we give a brief solution for paths.

3.7.1 Weighted coloured token swapping on paths

As mentioned above, we should never swap two tokens of the same colour. As noted by Yamanaka et al. [149], for the case of paths, this constraint imposes a unique assignment of tokens to vertices: the i^{th} token of colour c along the path must be assigned to the the i^{th} vertex of colour c.

It remains to solve the weighted token swapping problem on paths. As in the unweighted case, the required swaps correspond precisely to the inversions, i.e., the pairs of tokens t, t' whose order in the initial token placement differs from their order in the final token placement. The minimum weight of a swap sequence is the sum, over all inversions t, t' of $w(t) + w(t')$.

3.7.2 Weighted coloured token swapping on stars

In this section we give a polynomial time algorithm for the weighted coloured token swapping problem on a star. As announced above, we will show that weights and colours can be dealt with separately.

Weighted token swapping on a star

The algorithm described in this section is an example of an optimal polynomial-time token swapping algorithm that must at times move a happy leaf token.

We assume that every token has a distinct colour so we know exactly which vertex every token must move to. Each token t has a weight $w(t)$ and the cost of swapping tokens t and t' is $w(t) + w(t')$. Let H and U be the sets of tokens initially on the happy and unhappy leaves, respectively. Let A, the set of *active* tokens, be all tokens except those in H, i.e., A is U plus the token at the center vertex.

In the token permutation, the cycle that contains the token at the center vertex of the star will be called the *unlocked cycle*, and all other cycles will be called *locked cycles*. Using this terminology, Lemma 27 states that the optimum number of swaps to solve the unweighted token swapping problem is $n_U + \ell$, where $n_U = |U|$ and ℓ is the number of non-trivial locked cycles. The intuition for the lemma, and the reason for our terminology, is that every locked cycle must be 'unlocked' by an external token, introducing one extra swap per locked cycle.

The number of swaps performed in the weighted case must be at least $n_U + \ell$ and we will show that an optimum solution uses either this lower bound or two extra swaps. The idea is the following: each of the locked cycles must be unlocked by some other token, and we want to use the cheapest possible token for this. Either we will use an active token and perform $n_U + \ell$ swaps, or we will introduce two extra swaps that bring and return a globally cheapest token from an initially happy leaf to the star center and use this token to unlock all the locked cycles.

Notation. The following notation will be used throughout Section 3.7. Let X be the unlocked cycle. Let x be a minimum weight token in X, a be a minimum weight token in A, and h be a minimum weight token in H (h might not exist if there are no happy leaves). Observe that $w(a) \leq w(x)$. As above, let ℓ denote the number of non-trivial locked cycles in the input token permutation. Finally, let $d(t)$ be the distance of token t from its home and let $D_w = \sum_{\text{token } t} w(t)d(t)$. Observe that D_w is a lower bound on the cost of weighted token swapping.

Before presenting the algorithm, we give an alternative formula for D_w. We will use this in the forthcoming section on weighted coloured stars. Also, it implies that in the case of unit weights, $D_w = 2n_U$, which will clarify how the present algorithm generalizes the unweighted case. For vertex v, recall our notation that $s^{-1}(v)$ is the initial token at v, and $f^{-1}(v)$ is the final token at v. Thus, a leaf vertex v is happy if and only if $s^{-1}(v) = f^{-1}(v)$.

Claim 28. $D_w = \sum\{w(s^{-1}(v)) + w(f^{-1}(v)) : v$ is an unhappy leaf $\}$.

Proof. If t is an unhomed token whose initial and final vertices are both leaves, then it contributes $2w(t)$ to both sides of the equation. If t is a token whose initial vertex is the center vertex and whose final vertex is a leaf, then it contributes $w(t)$ to both sides. Similarly, a token whose initial vertex is a leaf and whose final vertex is the center, contributes $w(t)$ to both sides. Finally, a token that is home contributes 0 to both sides. □

Corollary 29. *When the weights are all 1, $D_w = 2n_U$.*

We now describe the algorithm for weighted token swapping on a star. The algorithm uses the best of three possible strategies, all of which begin the same way:

1. Stategy 1. Begin solving the unlocked cycle X by repeatedly swapping the token from the star center to its home until the token x is on the star center. Next, use x to unlock and solve all the locked cycles. Finally, complete solving X. The total weight is $D_w + 2w(x)\ell$.

2. Strategy 2. This strategy only applies when $w(a) < w(x)$, in which case $a \in U \backslash X$. Begin solving the unlocked cycle X by repeatedly swapping the token from the star center to its home until the token x is on the star center. Then swap x with a. Suppose a was in the locked cycle L. Use a to unlock and solve all the other locked cycles, leaving tokens of X and $L \setminus \{a\}$ fixed. Then use a to solve cycle

L, which will return x to the center token. Finally, complete solving X. The effect is that one locked cycle is unlocked by x at a cost of $2w(x)$ and $\ell - 1$ cycles are unlocked by a at a cost of $2w(a)(\ell-1)$, for a total cost of $D_w+2w(x)+2w(a)(\ell-1)$.

3. Strategy 3. This strategy only applies when h exists. Begin solving the unlocked cycle X by repeatedly swapping the token from the star center to its home until the token x is on the star center. Then swap x with h. Use h to unlock and solve all the locked cycles, leaving tokens of X fixed. Then swap h and x. Finally, complete solving X. The total weight is $D_w + 2w(x) + 2w(h) + 2w(h)\ell = D_w + 2w(x) + 2w(h)(\ell + 1)$.

To decide between the strategies we find the minimum of $w(x)(\ell - 1)$, $w(a)(\ell - 1)$, $w(h)(\ell + 1)$ and use the corresponding strategy 1, 2, or 3, respectively, achieving a total weight of $D_w + 2w(x) + 2\min\{w(a)(\ell - 1), w(h)(\ell + 1)\}$.

Theorem 30. *The above algorithm finds a minimum weight swap sequence and the weight of the swap sequence is:*

$$D_w + 2w(x) + 2\min\{w(a)(\ell - 1), w(h)(\ell + 1)\}.$$

Observe that in the case of unit weights, $D_w = 2n_U$ by Corollary 29, so the theorem says that the minimum number of token moves is $2n_U + 2 + 2(\ell - 1) = 2n_U + 2\ell$, i.e., the number of swaps is $n_U + \ell$, which matches what we know for the unweighted case.

To prove the theorem, we will need the following result about the unweighted star.

Lemma 31. *Any swap sequence on an unweighted star that moves a happy leaf does at least two more swaps than an optimal swap sequence.*

Proof. By Lemma 27, solving the unweighted problem on a star optimally takes $n_U + \ell$ swaps, where n_U is the number of unhappy leaves and ℓ is the number of non-trivial locked cycles. It suffices to check that after swapping a happy token with the center token the value given by the formula is increased by one. Indeed, the number of non-trivial locked cycles stays the same and the number of unhomed leaves increases by one, hence, the net change is +1. □

We now prove Theorem 30.

Proof. The swap sequence found by the algorithm realizes the formula given in the theorem. It remains to show that the formula provides a lower bound on the weight of any swap sequence.

To reach its home, each token t must contribute weight at least $w(t)d(t)$, for a total over all tokens of D_w. If $\ell = 0$ then $U - X$ is empty so $w(a) = w(x)$ and the formula evaluates to D_w, which is a lower bound. Assume from now on that $\ell \geq 1$. In addition to the moves accounted for in D_w, there must be at least 2ℓ other token moves, two for each locked cycle. Furthermore, there must be a first move that swaps some token t of X with a token outside X. This swap can only happen when t is at the center vertex. Since t will then be unhomed, there must be a move that returns token t back to the center vertex. The minimum weight for each of these moves is $w(x)$, and this provides the term $2w(x)$ in the lower bound. Subtracting these two moves from the required 2ℓ moves leaves $2(\ell - 1)$ moves still to be accounted for.

We now consider two cases depending whether a token of H is moved or not. If no token of H is moved in the swap sequence, then the best we can do for the remaining $2(\ell - 1)$ moves is to use a minimum weight token from A, so the weight is at least $D_w + 2w(x) + 2(\ell - 1)w(a)$.

Next, consider swap sequences that move a token of H. By Lemma 31, the sequence must do at least two extra swaps, i.e., at least 4 extra token moves. Thus the number of moves (beyond those for D_w) is at least $2\ell + 4$. As argued above, we need two moves for a token of X, costing at least $w(x)$ each. We also need two moves for a token of H (to leave its home and then return) costing at least $w(h)$ each. This leaves 2ℓ further moves. This gives weight at least $D_w + 2w(x) + 2w(h) + 2\ell \min\{w(a), w(h)\}$. If $w(h) \geq w(a)$ then this lower bound is higher than the previous ones and becomes irrelevant. If $w(h) < w(a)$, then the bound becomes $D_w + 2w(x) + 2w(h) + 2\ell w(h) = D_w + 2w(x) + 2w(h)(\ell + 1)$.

Thus, combining these possibilities, we have a lower bound of $D_w + 2w(x) + 2\min\{w(a)(\ell - 1), w(h)(\ell + 1)\}$, which completes the proof. $\square$

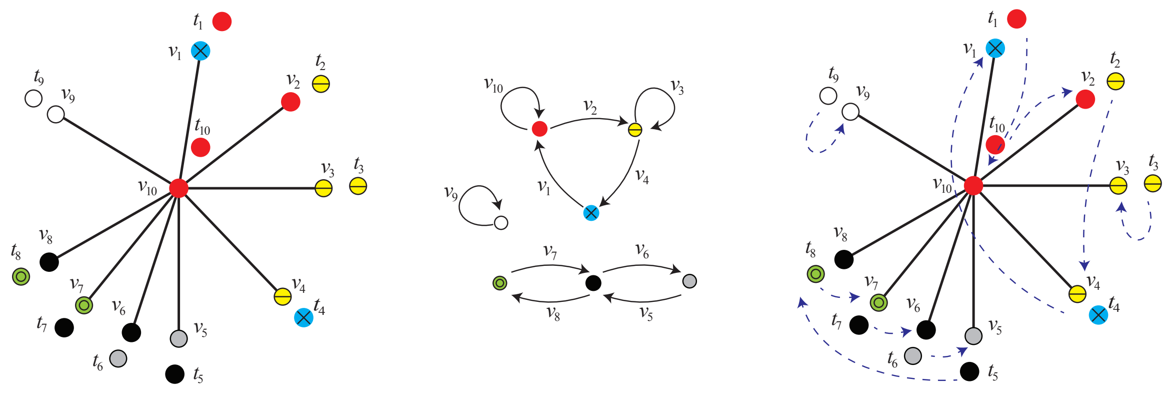


Figure 3.7: Left: an input for coloured token swapping on a star. The token at a vertex is drawn as a disc near the vertex. A token must move to a vertex of the same colour. Middle: the multi-graph G with edges labelled by the corresponding vertex of the star. There are 3 loops but one of them corresponds to the center vertex of the star, so $\lambda = 2$. There are 3 connected components, but one is trivial, and one contains the edge corresponding to the center vertex so $\kappa = 1$. Right: a token-vertex assignment (shown by the dashed arrows) that minimizes $n_U + \ell$. One may also observe that assigning token t_{10} to the center vertex v_{10} or token t_7 to vertex v_8 are both sub-optimal.

Coloured token swapping on a star

Recall that in the coloured token swapping problem, tokens and vertices are assigned colours, possibly with multiple tokens and vertices sharing the same colour, and the aim is to move the tokens to vertices of corresponding colours using a minimum number of swaps.

Our algorithm will find a *token-vertex assignment* that maps each token t of colour c to a vertex v of colour c, with the interpretation that token t should move to vertex v in the token swapping problem. Such an assignment yields a standard token-swapping problem which, by Lemma 27, requires $n_U + \ell$ swaps, where n_U is the number of unhappy leaves and ℓ is the number of non-trivial locked cycles. Thus, we want a token-vertex assignment that minimizes $n_U + \ell$. Note that minimizing n_U is the same as maximizing n_H, the number of happy leaves since $n_U = n - 1 - n_H$, where n is the number of vertices in the star.

We will find an optimum token-vertex assignment using an auxiliary multigraph G that has a vertex for each colour and one edge for each vertex of the star: for a vertex of colour c with an initial token of colour d, add a directed edge from c to d in G. In case $c = d$ this edge is a loop. See Figure 3.7 for an example.

Let λ be the number of *leaf loops* of G—loops corresponding to leaf vertices of the star. Any leaf loop corresponding to leaf vertex v and token t can be turned into a happy leaf by assigning token t to vertex v. This maximizes the number of happy leaves, n_H.

In the input, the number of vertices of colour c is equal to the number of tokens of colour c. Thus, each vertex of G has in-degree equal to out-degree, which implies that any connected component in G is strongly connected and has a directed Eulerian tour. We call a connected component *trivial* if it has one vertex. Let κ be the number of non-trivial connected components of G not counting the component that contains the edge corresponding to the center vertex of the star.

The algorithm for coloured token swapping on a star is as follows:

1. Find a token-vertex assignment:
   (a) Construct the multigraph G.
   (b) For each of the λ leaf-loops, assign its token to its vertex.
   (c) Remove the leaf-loops from G to obtain G'. Observe that κ is unchanged, and G' is still Eulerian. For each connected component of G' find an Eulerian tour that traverses all the edges of the component. Convert each Eulerian tour to a token-vertex assignment as follows: Suppose the edges of the tour are labelled by vertices $v_1, v_2, \ldots, v_b$ (we are freely re-labelling vertices to ease the notation), and suppose that the edge of G labelled v_i goes from colour c_{i-1} to colour c_i (subscript addition modulo b). Then vertex v_i has colour c_{i-1} and the colour of its initial token, say t_i, is c_i. The next edge in the tour corresponds to vertex v_{i+1} of colour c_i. Assign token t_i to vertex v_{i+1}. Note that both have colour c_i. This assignment is well-defined since the edges of the walk correspond to distinct vertices with distinct initial tokens. Note that this token-vertex assignment introduces a cycle $t_1, t_2, \ldots t_b$ in the corresponding token permutation.
2. Solve the (un-coloured) token swapping problem determined by the computed token- vertex assignment.

This algorithm produces a token-vertex assignment with λ happy leaves, and κ non-trivial locked cycles, one for each non-trivial connected component of G' except the component that contains the edge corresponding to the center vertex of the star. In other words, $n_H = \lambda$, $n_U = n - 1 - \lambda$ and $\ell = \kappa$. Thus the number of swaps is $(n - 1 - \lambda) + \kappa$ by Lemma 27.

Our goal in the remainder of this subsection is to prove that the algorithm uses the minimum number of swaps which means showing that any token-vertex assignment results in at least $(n - 1 - \lambda) + \kappa$ swaps. The following lemma will help us do that. (Note: the third statement in the lemma will be useful in the next section.)

Lemma 32. *Any token-vertex assignment T has the following properties:*

1. *T has at most λ happy leaves.*
2. *T has at least κ non-trivial locked cycles.*
3. *If T has λ happy leaves then the tokens in the unlocked cycle of T are a subset of X_A, where X_A is the set of tokens that are in the unlocked cycle resulting from the above algorithm.*

Proof. 1. Happy leaves only arise from leaf loops so T has at most λ happy leaves.

2. The token permutation corresponding to T can be expressed as a set $\mathcal{C}$ of cycles. We claim that each cycle $C \in \mathcal{C}$ corresponds to a closed walk $\bar{C}$ of the same size in G and that every edge of G is in $\bar{C}$ for some $C \in \mathcal{C}$. This will prove property 2, because it implies that we need at least one cycle for each connected component in G, and more precisely, that we need at least one non-trivial locked cycle for each of the components counted in κ.

Consider an edge of G, say the edge corresponding to the vertex whose initial token is t_1. Token t_1 appears in some cycle $C \in \mathcal{C}$, say $(t_1, t_2, \ldots, t_b)$. (We are freely re-naming tokens, vertices, and colours in this proof.) Suppose token t_i has colour c_i and is initially at vertex v_i. Then the cycle moves token t_i to vertex v_{i+1} (subscript addition modulo b). Since the token-vertex assignment respects the colours, vertex v_{i+1} has colour c_i. Also, vertex v_{i+1} has initial token t_{i+1} of colour c_{i+1}. Thus there is a corresponding edge c_i, c_{i+1} in G. Therefore, the cycle corresponds to a closed walk in G. Also, this closed walk uses the edge we began with, the one whose initial token is t_1.

3. The unlocked cycle of T is the one that contains the token t_c initially on the center vertex u. By the argument above, the tokens in the unlocked cycle must come from the connected component of G that contains the edge labelled with u. This set of tokens consists of X_A together with some tokens of leaf-loops. But if T has λ happy leaves, then all the leaf-loops have been turned into happy leaves, so the set of tokens is reduced to X_A. Thus, the tokens of the unlocked cycle are a subset of X_A. □

We are now ready to prove that the algorithm is optimal:

Theorem 33. *The above algorithm uses $(n-1-\lambda)+\kappa$ swaps and this is the minimum possible.*

Proof. As already stated, the algorithm uses $(n-1-\lambda)+\kappa$ swaps.

By Lemma 32 any other token-vertex assignment results in at most λ happy leaves, i.e. at least $n-1-\lambda$ unhappy leaves, and at least κ non-trivial locked cycles, and therefore, by Lemma 27, at least $(n-1-\lambda)+\kappa$ swaps. □

Weighted coloured token swapping on a star

Our algorithm for weighted coloured token swapping on a star is as follows:

1. Ignore the weights and find a token-vertex assignment as in Step 1 of the algorithm in the previous section.

2. Using this token-vertex assignment T and the original token weights, run the algorithm for the (uncoloured) weighted star.

In order to show that this algorithm is correct, we will first show that any optimum token-vertex assignment must turn all leaf-loops into happy leaves. After that we only need to compare the solution found by the algorithm to solutions with this property.

Claim 34. Suppose T is a token-vertex assignment and there is a leaf-loop consisting of a leaf vertex v with token t such that both v and t have colour c, but the token-vertex assignment does not assign t to v. Then T is not optimum for the weighted problem.

Proof. By Theorem 30, the cost of T is

$$F(T) = D_w + 2w(x) + 2\min\{w(a)(\ell - 1), w(h)(\ell + 1)\},$$

where $D_w, w(x), w(a), w(h)$, and ℓ depend on T. We will construct a new token-vertex assignment T' that assigns t to v and has $F(T') < F(T)$.

Since t is not assigned to v, t must be part of some non-trivial cycle C in the token permutation determined by T. Suppose that the cycle C contains tokens p, t, q in that order (possibly $p = q$), with initial vertices $s(p), s(t){=}v, s(q)$, respectively. Define a new token-vertex assignment T' that assigns t to v, i.e., v becomes a happy leaf, and shortcuts the rest of C by assigning token p to vertex $s(q)$. This is valid because token p and vertex $s(q)$ both have colour c, the same as t. The new cycle C' is formed by deleting t from C. We will compare $F(T)$ and $F(T')$ by looking at the quantities $D_w, w(x), w(a), w(h)$ and ℓ.

First of all, no leaf becomes unhappy, so no token leaves H and $w(h)$ does not increase. Furthermore, v becomes happy, so by Claim 28, D_w decreases by at least $2w(t)$.

Next we show that $w(x)$ does not increase. That would only happen if t leaves the set X. Then C must be the unlocked cycle. Since t is at a leaf vertex, the token from the center vertex remains in C', so C' is the new unlocked cycle. Furthermore, token p, which is a ‘twin’ of t in the sense that it has the same colour and weight, remains in C', so $w(x)$ remains the same.

Finally we must consider ℓ and $w(a)$. Here we will separate out one special case—when $|C| = 2$ and C exchanges two leaf tokens, in which case C' becomes a trivial locked cycle. If we are not in the special case then either C' is a non-trivial locked cycle, or C' is the unlocked cycle. In either case C has the same status, so ℓ is unchanged and t's twin p remains in the active set A so $w(a)$ does not increase. Thus $F(T') < F(T)$ when we are not in the special case.

It remains to consider the special case when C exchanges two leaf tokens. Then C was a non-trivial locked cycle, but C' is a trivial locked cycle. Thus ℓ decreases by 1. Furthermore, by Claim 28, D_w decreases by at least $4w(t)$ since two leaves become happy. If $w(a)$ does not increase then we are fine. If it does increase then

$w(a) = w(t)$ and t was the minimum weight element in A. Because we are in the special case, both t and its twin token p have left A and joined H. If $F(T)$ is determined by $w(h)(\ell+1)$ we are again fine. Hence we only need to provide an additional argument if $F(T) = D_w + 2w(x) + 2w(a)(\ell - 1)$. Since we now have a token of weight $w(a) = w(t)$ in H, Strategy 3 gives a swap sequence for T' of weight at most $(D_w - 4w(t)) + 2w(x) + 2w(t)(\ell) = D_w - 2w(t) + 2w(x) + 2w(t)(\ell - 1)$. Thus $F(T') < F(T)$ even in the special case. □

With this claim in hand, we are ready to prove that the algorithm is correct.

Theorem 35. *The above algorithm solves the weighted coloured token swapping problem on a star optimally.*

Proof. By Theorem 30, the cost of a token-vertex assignment T is

$$F(T) = D_w + 2w(x) + 2\min\{w(a)(\ell - 1), w(h)(\ell + 1)\},$$

where $D_w, w(x), w(a), w(h)$, and ℓ depend on T.

We will compare the cost of a token-vertex assignment T_A found by the algorithm to an optimum token-vertex assignment T_{OPT}. By Claim 34, T_{OPT} turns all leaf-loops into happy leaves, so it has λ happy leaves. The algorithm does the same, so T_A and T_{OPT} have the same set H of tokens on happy leaves, and the same set U of tokens on unhappy leaves. This implies that $w(a)$ and $w(h)$ are the same for T_A and T_{OPT}.

Next, we claim that D_w is the same for T_A and T_{OPT}. This follows directly from Claim 28 since the set of unhappy leaves is the same.

It remains to compare ℓ (the number of non-trivial locked cycles) and $w(x)$ between T_A and T_{OPT}. Both values should be as small as possible in T_{OPT}. The algorithm achieves $\ell = \kappa$ and $w(x) = \min\{w(t) : t \in X_A\}$, where X_A is the set of tokens in the unlocked cycle of T_A. By Lemma 32(2) T_{OPT} has at least κ non-trivial locked cycles. By Lemma 32(3), T_{OPT}'s set of tokens in the unlocked cycle is a subset of X_A (here we again use the fact that T_{OPT} has λ happy leaves). Thus T_A and T_{OPT} achieve the same values for ℓ and $w(x)$. This completes the proof that the algorithm achieves the minimum value of $F(T)$. □

4 Conclusions

This thesis presented results related to reconfiguration of triangulations of planar point sets and of token arrangements on graphs. Both topics were set and discussed in the greater context of the reconfiguration framework.

In the language of reconfiguration, our results establish a polynomial time checkable criterion, the Orbit Theorem, to determine reachability in the reconfiguration graph of edge-labelled triangulations with the labelled edge-flip operation. The presented proof of the Orbit Theorem, moreover, gives a polynomial time algorithm to find a reconfiguration sequence, if it exists. In token swapping, we have made partial progress towards the problem of shortest transformation in the reconfiguration graph of tokens on tree graphs using the swapping operation.

For open problems related to the general reconfiguration framework, we point the reader to [111] that provides a rich summary. One obvious quest is to unify results from different fields and identify general patterns.

We conclude the thesis with remarks and open problems specific to the two reconfiguration topics studied in this thesis: the reconfiguration of edge-labelled triangulations and the token swapping problem on trees.

4.1 Conclusions and Open Problems related to Orbit Theorem

We have characterized when two labelled triangulations of a set of n points belong to the same connected component of the labelled flip graph, and proved that the diameter of each connected component is bounded by $O(n^7)$. The following is a list of some open problems:

1. Reduce the gap between the upper bound, $O(n^7)$, and the best known lower bound of $O(n^3)$ [31] on the diameter of a component of the labelled flip graph.

2. We have studied the case where each edge in a triangulation has a unique label, and given a bound of $O(n^7)$ on the diameter of a component of the labelled flip graph. The case where edges are unlabelled can be viewed as the case where every edge has the same label—in this case the bound becomes $O(n^2)$. A unifying scenario, suggested by Giuseppe Liotta at Symposium on Computational Geometry'17, is when the edges have labels and labels may appear on more than one edge. Is there a bound on the diameter of connected components of the flip graph that depends on the number of labels, or on the maximum number of edges with the same label?

   We note that the existential part of Orbit Theorem applies to the unifying scenario. Without loss of generality, suppose that the two given labelled triangulations are $\mathcal{T}_1 = (T, \ell_1)$ and $\mathcal{T}_2 = (T, \ell_2)$ and denote the number of edges in orbit i having label l by $|\mathcal{T}_1|_{li}$ and $|\mathcal{T}_2|_{li}$, respectively. Clearly, if there is i and l such that $|\mathcal{T}_1|_{li} \neq |\mathcal{T}_2|_{li}$, i.e., some orbit has different number of edges labelled l in $\mathcal{T}_1$ than in $\mathcal{T}_2$, then it is not possible to reconfigure $\mathcal{T}_1$ into $\mathcal{T}_2$. Conversely, if for all orbits i and labels $l,$ $|\mathcal{T}_1|_{li} = |\mathcal{T}_2|_{li}$, i.e., within each orbit the number of edges labelled l coincides between $\mathcal{T}_1$ and $\mathcal{T}_2$, then the reconfiguration is possible: Theorem 2 guarantees that two labels of edges in the same orbit can be swapped while fixing all other labels in the triangulation. Hence, the labels get to (any of) their target positions by inductively applying Theorem 2.

3. We did not analyze the run-time of our algorithms in the main text, and in particular the run-time of the algorithm on page 80. A crude bound is $O(n^8)$, with the bottleneck being the explicit construction in the proof of Lemma 4 of the double quadrilateral graph which has $O(n^4)$ vertices and thus $O(n^8)$ edges. This bound can surely be improved.

4.2 Conclusions and open problems related to token swapping on trees

Although we have not resolved the question of whether token swapping on a tree is in P or NP-complete, we have identified a previously unexplored difficulty—namely that we must decide how and when to move tokens that are at happy leaves. This difficulty does not arise for the cases where poly-time algorithms are known, specifically, paths, stars and brooms. We have demonstrated that even on rather uncomplicated graphs, like three paths meeting in a single vertex, moving the happy tokens/leaves can save a constant factor of swaps. In [24] we showed that any algorithm that fixes tokens at happy leaves cannot achieve better than a $\frac{4}{3}$ approximation factor, and that this lower bound rises to 2 for two of the three known 2-approximation algorithms, thus providing tight approximation factors for them.

Furthermore, we established a difference in complexity between general trees on the one hand, and paths and stars on the other hand, namely that weighted coloured token swapping is NP-hard for general trees, but poly-time for paths and stars.

We conclude with some open questions.

1. Is the token swapping problem on trees NP-complete? Is it in P? For hardness, a first step would be to show that the problem is NP-complete with either colours or weights (rather than both, as we proved).

2. Characterize the class of trees for which the Happy Leaf Conjecture holds for every token assignment. Certainly the tree should not have the 10-vertex tree of Figure 3.3 as a subtree.

3. Is there a polynomial time algorithm for token swapping on any tree for which the Happy Leaf Conjecture holds? This may not be easy, given the difficulty of correctly solving token swapping on such uncomplicated classes of graphs as brooms – where a broom is a star with a path attached, see [24].

4. Is there an approximation algorithm for token swapping on a tree with approximation factor better than 2? What is the exact approximation factor of Vaughan's algorithm? We conjecture that it is 2, perhaps even for the same example as used in proving that the happy swap and the cycle algorithms do not have an ap-

proximation factor less than 2, see [24]. The proof seems more elusive because a token can stray further from the path between its initial and target vertices.

5. The example which defeats all algorithms that fix happy leaves, consists of a star joined to two paths. Such a *two-tailed star* is like a broom with an extra handle. We conjecture that there is a polynomial time algorithm for token-swapping on two-tailed stars. This would be a starting point towards solving token swapping when happy leaves must be swapped.

6. For general graphs there is a 4-approximation algorithm [106] for token swapping. Is the approximation factor 4 tight?

Bibliography

[1] Manuel Abellanas, Prosenjit Bose, Alfredo García, Ferran Hurtado, Pedro Ramos, Eduardo Rivera-Campo, and Javier Tejel. On local transformations in plane geometric graphs embedded on small grids. *Computational Geometry*, 39(2):65–77, 2008.

[2] Oswin Aichholzer, Victor Alvarez, Thomas Hackl, Alexander Pilz, Bettina Speckmann, and Birgit Vogtenhuber. An Improved Lower Bound on the Minimum Number of Triangulations. In Sándor Fekete and Anna Lubiw, editors, *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[3] Oswin Aichholzer, Andrei Asinowski, and Tillmann Miltzow. Disjoint compatibility graph of non-crossing matchings of points in convex position. *The electronic journal of combinatorics*, 22(1):1–53, 2015. 22:P1.65.

[4] Oswin Aichholzer, Franz Aurenhammer, and Ferran Hurtado. Sequences of spanning trees and a fixed tree theorem. *Computational Geometry*, 21(1):3 – 20, 2002. Sixteenth European Workshop on Computational Geometry (EuroCG-2000).

[5] Oswin Aichholzer, Franz Aurenhammer, Hannes Krasser, and Peter Brass. Pseudotriangulations from surfaces and a novel type of edge flip. *SIAM J. Comput.*, 32(6):1621–1653, June 2003.

[6] Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport,

Shakhar Smorodinsky, Diane Souvaine, Jorge Urrutia, and David R. Wood. Compatible geometric matchings. *Computational Geometry*, 42(6):617 – 626, 2009.

[7] Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015.

[8] Oswin Aichholzer and Klaus Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Computational Geometry*, 37(3):155 – 161, 2007. Special Issue on the 20th European Workshop on Computational Geometry.

[9] Sheldon B. Akers and Balakrishnan Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, 1989.

[10] Greg Aloupis, Luis Barba, Stefan Langerman, and Diane L. Souvaine. Bichromatic compatible matchings. *Computational Geometry*, 48(8):622 – 633, 2015.

[11] Greg Aloupis, Prosenjit Bose, and Pat Morin. Reconfiguring triangulations with edge flips and point moves. *Algorithmica*, 47(4):367–378, Apr 2007.

[12] Amihood Amir, Tzvika Hartman, Oren Kapah, Avivit Levy, and Ely Porat. On the cost of interchange rearrangement in strings. *SIAM J. Comput.*, 39(4):1444–1461, 2009.

[13] Amihood Amir and Benny Porat. On the hardness of optimal vertex relabeling and restricted vertex relabeling. In *Annual Symposium on Combinatorial Pattern Matching*, volume 9133 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2015.

[14] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.

[15] Gabriela Araujo-Pardo, Isabel Hubard, Deborah Oliveros, and Egon Schulte. Colorful associahedra and cyclohedra. *Journal of Combinatorial Theory, Series A*, 129:122–141, 2015.

[16] Vincenzo Auletta, Angelo Monti, Mimmo Parente, and Pino Persiano. A linear-time algorithm for the feasibility of pebble motion on trees. *Algorithmica*, 23(3):223–245, 1999.

[17] Vincenzo Auletta and Pino Persiano. Optimal pebble motion on a tree. *Information and Computation*, 165(1):42–68, 2001.

[18] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21 – 46, 1996. First International Colloquium on Graphs and Optimization.

[19] Lajos Balcza. On inversions and cycles in permutations. *Periodica Polytechnica Civil Engineering*, 36(4):369–374, 1992.

[20] Sergey Bereg. Transforming pseudo-triangulations. *Information Processing Letters*, 90(3):141 – 145, 2004.

[21] Marshall Bern, Herbert Edelsbrunner, David Eppstein, Scott Mitchell, and Tio Seng Tan. Edge insertion for optimal triangulations. *Discrete & Computational Geometry*, 10(1):47–65, 1993.

[22] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. World Scientific, 1992.

[23] R. H. Bing. Some aspects of the topology of 3-manifolds related to the Poincaré conjecture. In *Lectures on modern mathematics, Vol. II*, pages 93–128. Wiley, New York, 1964.

[24] Ahmad Biniaz, Kshitij Jain, Anna Lubiw, Zuzana Masárová, Tillmann Miltzow, Debajyoti Mondal, Anurag Murty Naredla, Josef Tkadlec, and Alexi Turcotte. Token swapping on trees. *CoRR*, abs/1903.06981, 2019.

[25] Ahmad Biniaz, Anil Maheshwari, and Michiel Smid. Flip distance to some plane configurations. *Computational Geometry*, 81:12 – 21, 2019.

[26] Anders Björner. Topological methods. *Handbook of combinatorics*, 2:1819–1872, 1995.

[27] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2nd edition, 1999.

[28] Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. Complexity of token swapping and its variants. *Algorithmica*, pages 1–27, 2017.

[29] Prosenjit Bose, Jurek Czyzowicz, Zhicheng Gao, Pat Morin, and David R Wood. Simultaneous diagonal flips in plane triangulations. *Journal of Graph Theory*, 54(4):307–330, 2007.

[30] Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry Theory and Applications*, 42(1):60–80, 2009.

[31] Prosenjit Bose, Anna Lubiw, Vinayak Pathak, and Sander Verdonschot. Flipping edge-labelled triangulations. *Computational Geometry*, 68:309–326, 2018.

[32] Prosenjit Bose and Sander Verdonschot. *A History of Flips in Combinatorial Triangulations*, pages 29–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[33] Prosenjit Bose and Sander Verdonschot. Flips in edge-labelled pseudo-triangulations. *Computational Geometry*, 60:45–54, 2017.

[34] Hervé Brönnimann, Lutz Kettner, Michel Pocchiola, and Jack Snoeyink. Counting and enumerating pointed pseudotriangulations with the greedy flip algorithm. *SIAM Journal on Computing*, 36(3):721–739, 2006.

[35] John L. Bryant. Piecewise linear topology. In R.J. Daverman and R.B. Sher, editors, *Handbook of Geometric Topology*, pages 219 – 259. North-Holland, Amsterdam, 2001.

[36] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Pancake flipping is hard. *Journal of Computer and System Sciences*, 81(8):1556–1574, 2015.

[37] Gruia Călinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. *SIAM Journal on Discrete Mathematics*, 22(1):124–138, 2008.

[38] Javier Cano, José-Miguel Díaz-Báñez, Clemens Huemer, and Jorge Urrutia. The edge rotation graph. *Graphs and Combinatorics*, 29(5):1207–1219, 2013.

[39] Jean Cardinal, Michael Hoffmann, Vincent Kusters, Csaba D. Tóth, and Manuel Wettstein. Arc diagrams, flip distances, and Hamiltonian triangulations. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[40] Arthur Cayley. LXXVII. Note on the theory of permutations. *Philosophical Magazine Series 3*, 34(232):527–529, 1849.

[41] Arthur Cayley. A theorem on trees. *Quart. J. Math.*, 23:376–378, 1889.

[42] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991.

[43] Siu-Wing Cheng, Tamal K. Dey, and Jonathan Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.

[44] L. Paul Chew. Constrained Delaunay triangulations. *Algorithmica*, 4(1-4):97–108, 1989.

[45] Bhadrachalam Chitturi. Upper bounds for sorting permutations with a transposition tree. *Discrete Mathematics, Algorithms and Applications*, 5(01):1350003, 2013.

[46] Gopal Danaraj and Victor Klee. Shellings of spheres and polytopes. *Duke Mathematical Journal*, 41(2):443–451, 1974.

[47] Mark de Berg, Bart M.P. Jansen, and Debankur Mukherjee. Independent-set reconfiguration thresholds of hereditary graph classes. *Discrete Applied Mathematics*, 250:165 – 182, 2018.

[48] Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[49] Erik D. Demaine and Robert A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In Michael H. Albert and Richard J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

[50] Satyan L. Devadoss and Joseph O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

[51] Adrian Dumitrescu, André Schulz, Adam Sheffer, and Csaba D. Tóth. Bounds on the maximum multiplicity of some common geometric graphs. *SIAM Journal on Discrete Mathematics*, 27(2):802–826, 2013.

[52] N. Dyn, I. Goren, and S. Rippa. Transforming triangulations in polygonal domains. *Computer Aided Geometric Design*, 10:531–536, 1993.

[53] Paul H. Edelman. On inversions and cycles in permutations. *European Journal of Combinatorics*, 8(3):269–279, 1987.

[54] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, Cambridge, 2001.

[55] Herbert Edelsbrunner and Tiow Seng Tan. A quadratic time algorithm for the minmax length triangulation. *SIAM Journal on Computing*, 22(3):527–551, 1993.

[56] Herbert Edelsbrunner, Tiow Seng Tan, and Roman Waupotitsch. An o(nˆ2\logn) time algorithm for the minmax angle triangulation. *SIAM Journal on Scientific and Statistical Computing*, 13(4):994–1008, 1992.

[57] David Eppstein. Happy endings for flip graphs. *Journal of Computational Geometry*, 1(1):3–28, 2010.

[58] David Eppstein. Regular labelings and geometric structures. In *Proceedings of the 22nd Canadian Conference on Computational Geometry (CCCG 2010)*, pages 125–130, 2010.

[59] David Eppstein. Counting polygon triangulations is hard. In *Proceedings of the 35th Annual Symposium on Computational Geometry (SoCG 2019), June 18-21, 2019, Portland, Oregon, USA*, pages 33:1–33:17, 2019.

[60] Jérémy Espinas, Raphaëlle Chaine, and Pierre-Marie Gandoin. Practical reduction of edge flip sequences in two-dimensional triangulations. *arXiv preprint arXiv:1310.2586*, 2013.

[61] Ruy Fabila-Monroy, David Flores-Peñaloza, Clemens Huemer, Ferran Hurtado, Jorge Urrutia, and David R. Wood. Token graphs. *Graphs and Combinatorics*, 28(3):365–380, 2012.

[62] Klaus-Tycho Foerster, Linus Groner, Torsten Hoefler, Michael Koenig, Sascha Schmid, and Roger Wattenhofer. Multi-agent pathfinding with n agents on graphs with n vertices: Combinatorial classification and tight algorithmic bounds. In *International Conference on Algorithms and Complexity (CIAC)*, pages 247–259. Springer, 2017.

[63] Fabrizio Frati. A lower bound on the diameter of the flip graph. *The Electronic Journal of Combinatorics*, 24(1):1–6, 2017.

[64] Jerôme Galtier, Ferran Hurtado, Marc Noy, Stéphane Pérennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *International Journal of Computational Geometry & Applications*, 13(02):113–133, 2003.

[65] Ashwin Ganesan. An efficient algorithm for the diameter of Cayley graphs generated by transposition trees. *International Journal of Applied Mathematics*, 42(4), 2012. Also arXiv preprint arXiv:1202.5888.

[66] Alfredo García, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Computational Geometry*, 16(4):211 – 221, 2000.

[67] Oded Goldreich. Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 1–5. Springer, 2011.

[68] Gilad Goraly and Refael Hassin. Multi-color pebble motion on graphs. *Algorithmica*, 58(3):610–636, 2010.

[69] Daniel Graf. How to sort by walking and swapping on paths and trees. *Algorithmica*, 78(4):1151–1181, 2017.

[70] Sabine Hanke. Ebene Triangulationen. Master's thesis, Institut für Informatik, University of Freiburg, 1994.

[71] Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.

[72] Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2000.

[73] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1):72 – 96, 2005. Game Theory Meets Theoretical Computer Science.

[74] C. Hernando, F. Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18(3):517–532, Oct 2002.

[75] Carmen Hernando, Ferran Hurtado, Mercè Mora, and Eduardo Rivera-Campo. Grafos de árboles etiquetados y grafos de árboles geométricos etiquetados. *Proc. X Encuentros de Geometra Computacional*, pages 13–19, 2003. (in Spanish).

[76] M.C. Hernando, F. Hurtado, A. Márquez, M. Mora, and M. Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51 – 66, 1999. 13th European Workshop on Computational Geometry CG '97.

[77] Michael Hoffmann, Micha Sharir, Adam Sheffer, Csaba D. Tóth, and Emo Welzl. Counting plane graphs: Flippability and its applications. In Frank Dehne, John Iacono, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, pages 524–535, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[78] M.E. Houle, F. Hurtado, M. Noy, and E. Rivera-Campo. Graphs of triangulations and perfect matchings. *Graphs and Combinatorics*, 21(3):325–331, Sep 2005.

[79] J. F. P. Hudson. *Piecewise Linear Topology*. W. A. Benjamin, Inc., New York-Amsterdam, 1969.

[80] Clemens Huemer and Anna de Mier. Lower bounds on the maximum number of non-crossing acyclic graphs. *European Journal of Combinatorics*, 48:48 – 62, 2015. Selected Papers of EuroComb'13.

[81] Ferran Hurtado and Marc Noy. Graph of triangulations of a convex polygon and tree of triangulations. *Computational Geometry*, 13(3):179–188, 1999.

[82] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Parallel edge flipping. In *Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG 1998), McGill University, Montréal, Québec, Canada, August 10-12, 1998*, pages 26–27, 1998.

[83] Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[84] Mashhood Ishaque, Diane L. Souvaine, and Csaba D. Tóth. Disjoint compatible geometric matchings. *Discrete & Computational Geometry*, 49(1):89–131, Jan 2013.

[85] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12–14):1054–1065, 2011.

[86] Mark R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985.

[87] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[88] Jesse Johnson. Notes on Heegard splittings, 2019 (accessed December 16, 2019).

[89] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9 – 15, 2012.

[90] Iyad Kanj, Eric Sedgwick, and Ge Xia. Computing the flip distance between triangulations. *Discrete & Computational Geometry*, 58(2):313–344, 2017.

[91] Jun Kawahara, Toshiki Saitoh, and Ryo Yoshinaka. The time complexity of permutation routing via matching, token swapping and a variant. *arXiv preprint arXiv:1612.02948*, 2016.

[92] Jun Kawahara, Toshiki Saitoh, and Ryo Yoshinaka. The time complexity of the token swapping problem and its parallel variants. In *The 11th International Conference and Workshop on Algorithms and Computation (WALCOM 2017)*, pages 448–459. Springer, 2017.

[93] Dohan Kim. Sorting on graphs by adjacent swaps using permutation groups. *Computer Science Review*, 22:89–105, 2016.

[94] Donald Ervin Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Pearson Education, 1997.

[95] Daniel Kornhauser, Gary Miller, and Paul Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984)*, pages 241–250. IEEE Computer Society, 1984.

[96] Benjamin Kraft. Diameters of Cayley graphs generated by transposition trees. *Discrete Applied Mathematics*, 184:178–188, 2015.

[97] Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972.

[98] Charles L. Lawson. Software for C^1 surface interpolation. In *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

[99] Errol L. Lloyd. On triangulations of a set of points in the plane. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, pages 228–240. IEEE, 1977.

[100] Anna Lubiw, Zuzana Masárová, and Uli Wagner. A proof of the Orbit Conjecture for flipping edge-labelled triangulations. *Discrete & Computational Geometry*, 61(4):880–898, Jun 2019.

[101] Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015.

[102] Anna Lubiw and Vinayak Pathak. Reconfiguring ordered bases of a matroid. *arXiv:1612.00958*, 2016.

[103] Joan M. Lucas. The rotation graph of binary trees is Hamiltonian. *Journal of Algorithms*, 8(4):503–535, 1987.

[104] Jiří Matoušek. *Using the Borsuk-Ulam theorem: lectures on topological methods in combinatorics and geometry*. Springer Science & Business Media, 2003.

[105] M Mihail and U Vazirani. On the expansion of 0-1 polytopes. *Journal of Combinatorial Theory, Series B, to appear*, 1989.

[106] Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and hardness of token swapping. In *24th Annual European Symposium on Algorithms (ESA 2016), LIPIcs-Leibniz International Proceedings in Informatics*, volume 57. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[107] Michael Molloy, Bruce Reed, and William Steiger. On the mixing rate of the triangulation walk. In *DIMACS-AMS volume on Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 179–190. AMS, 1999.

[108] Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, May 2017.

[109] James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley Publishing Company, Menlo Park, CA, 1984.

[110] Torrie L. Nichols, Alexander Pilz, Csaba D. Tóth, and Ahad N. Zehmakan. Transition operations over plane trees. In Michael A. Bender, Martín Farach-Colton, and Miguel A. Mosteiro, editors, *LATIN 2018: Theoretical Informatics*, pages 835–848, Cham, 2018. Springer International Publishing.

[111] Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

[112] Takao Nishizeki and Norishige Chiba. *Planar Graphs: Theory and Algorithms*, volume 32. Elsevier, 1988.

[113] Jakob Nordström. Pebble games, proof complexity, and time-space trade-offs. *Logical Methods in Computer Science*, 9(3):15, 2013.

[114] David Orden and Francisco Santos. The polytope of non-crossing graphs on a planar point set. *Discrete & Computational Geometry*, 33(2):275–305, 2005.

[115] James Oxley. *Matroid Theory, Second Edition*. Oxford University Press, 2011.

[116] Igor Pak. Reduced decompositions of permutations in terms of star transpositions, generalized Catalan numbers and k-ary trees. *Discrete Mathematics*, 204(1-3):329–335, 1999.

[117] Christos H. Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994)*, pages 511–520. IEEE, 1994.

[118] Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014.

[119] Alexander Pilz. A note on the flip distance problem for edge-labeled triangulations. *arXiv preprint arXiv:1808.03126*, 2018.

[120] Frederick J. Portier and Theresa P. Vaughan. Whitney numbers of the second kind for the star poset. *European Journal of Combinatorics*, 11(3):277–288, 1990.

[121] Lionel Pournin. The diameter of associahedra. *Advances in Mathematics*, 259:13–42, 2014.

[122] Ke Qiu, Selim G. Akl, and Henk Meijer. On some properties and algorithms for the star and pancake interconnection networks. *Journal of Parallel and Distributed Computing*, 22(1):16–25, 1994.

[123] Daniel Ratner and Manfred Warmuth. The (n^2-1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.

[124] Andreas Razen. A lower bound for the transformation of compatible perfect matchings. *Proceedings of the 24th European Workshop on Computational Geometry (EuroCG 2008)*, pages 115–118, 2008.

[125] J. Scherphuis. Rotational puzzles on graphs. `http://www.jaapsch.net/puzzles/graphpuzz.htm`.

[126] Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 138–148, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[127] Herbert Seifert and William Threlfall. *A Textbook of Topology*, volume 89 of *Pure and Applied Mathematics*. Academic Press, 1980.

[128] Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *Electronic journal of combinatorics*, 18(1):70, 2011.

[129] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988.

[130] Daniel D. Sleator, Robert E. Trajan, and William P. Thurston. Short encodings of evolving structures. *SIAM Journal on Discrete Mathematics*, 5(3):428–450, 1992.

[131] John H. Smith. Factoring, into edge transpositions of a tree, permutations fixing a terminal vertex. *Journal of Combinatorial Theory, Series A*, 85(1):92–95, 1999.

[132] John H. Smith. Corrigendum: Corrigendum to factoring, into edge transpositions of a tree, permutations fixing a terminal vertex [J. Combin. Theory Ser. A 85 (1)(1999) 92-95]. *Journal of Combinatorial Theory Series A*, 118(2):726–727, 2011.

[133] John Stillwell. *Classical Topology and Combinatorial Group Theory*, volume 72 of *Graduate Texts in Mathematics*. Springer-Verlag, 2nd edition, 1993.

[134] I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, pages 443–453, Nov 2000.

[135] William T. Tutte. A theorem on planar graphs. *Transactions of the American Mathematical Society*, 82(1):99–116, 1956.

[136] William T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13(3):743–768, 1963.

[137] Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics*, 409:127–160, 2013.

[138] Tzvetalin S. Vassilev. *Optimal area triangulation*. PhD thesis, University of Saskatchewan Saskatoon, 2005.

[139] Theresa P. Vaughan. Bounds for the rank of a permutation on a tree. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 10:65–81, 1991.

[140] Theresa P. Vaughan. Factoring a permutation on a broom. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 30:129–148, 1999.

[141] Theresa P. Vaughan and Frederick J. Portier. An algorithm for the factorization of permutations on a tree. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 18:11–31, 1995.

[142] Klaus Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[143] Ling Wang, Satishkumar Subramanian, Shahram Latifi, and Pradip K. Srimani. Distance distribution of nodes in star graphs. *Applied Mathematics Letters*, 19(8):780–784, 2006.

[144] Richard M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974.

[145] Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *J. Comput. Syst. Sci.*, 93:1–10, 2014.

[146] Marcin Wrochna. The topology of solution spaces of combinatorial problems. 2018.

[147] Katsuhisa Yamanaka, Erik D. Demaine, Takashi Horiyama, Akitoshi Kawamura, Shin-Ichi Nakano, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Ryuhei Uehara, and Takeaki Uno. Sequentially swapping colored tokens on graphs. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *The 11th International Conference and Workshop on Algorithms and Computation (WALCOM 2017)*, volume 10167 of *Lecture Notes in Computer Science*, pages 435–447. Springer, 2017.

[148] Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. *Theoretical Computer Science*, 586:81–94, 2015.

[149] Katsuhisa Yamanaka, Takashi Horiyama, J. Mark Keil, David Kirkpatrick, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, and Yushi Uno. Swapping colored tokens on graphs. *Theoretical Computer Science*, 729:1–10, 2018.

[150] Katsuhisa Yamanaka, Takashi Horiyama, David Kirkpatrick, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, and Yushi Uno. Swapping colored tokens on graphs. In *Workshop on Algorithms and Data Structures (WADS 2015)*, pages 619–628. Springer, 2015.

[151] Chao Yang. Sliding puzzles and rotating puzzles on graphs. *Discrete Mathematics*, 311(14):1290–1294, 2011.

[152] Gaku Yasui, Kouta Abe, Katsuhisa Yamanaka, and Takashi Hirayama. Swapping labeled tokens on complete split graphs. *SIG Tech. Rep.*, 2015(14):1–4, 2015.

[153] Jingjin Yu. Intractability of optimal multirobot path planning on planar graphs. *IEEE Robotics and Automation Letters*, 1(1):33–40, 2016.

[154] Jingjin Yu and Steven M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

[155] Jingjin Yu and Daniela Rus. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In *Algorithmic foundations of robotics XI*, pages 729–746. Springer, 2015.