Hamiltonian Cycle and Related Problems: Vertex-Breaking, Grid Graphs, and Rubik's Cubes

by

Mikhail Rudoy

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2017

Author .....
Department of Electrical Engineering and Computer Science
May 26, 2017

Certified by .....
Erik Demaine
Professor
Thesis Supervisor

Accepted by .....
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

THIS PAGE INTENTIONALLY LEFT BLANK

Hamiltonian Cycle and Related Problems:
Vertex-Breaking, Grid Graphs, and Rubik's Cubes

by

Mikhail Rudoy

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2017, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, we analyze the computational complexity of several problems related to the Hamiltonian Cycle problem.

We begin by introducing a new problem, which we call Tree-Residue Vertex-Breaking (TRVB). Given a multigraph G some of whose vertices are marked "breakable," TRVB asks whether it is possible to convert G into a tree via a sequence of applications of the vertex-breaking operation: disconnecting the edges at a degree-k breakable vertex by replacing that vertex with k degree-1 vertices. We consider the special cases of TRVB with any combination of the following additional constraints: G must be planar, G must be a simple graph, the degree of every breakable vertex must belong to an allowed list B, and the degree of every unbreakable vertex must belong to an allowed list U. We fully characterize these variants of TRVB as polynomially solvable or NP-complete. The TRVB problem is useful when analyzing the complexity of what could be called single-traversal problems, where some space (i.e., a configuration graph or a grid) must be traversed in a single path or cycle subject to local constraints. When proving such a problem NP-hard, a reduction from TRVB can often be used as a simpler alternative to reducing from a hard variant of Hamiltonian Cycle.

Next, we analyze several variants of the Hamiltonian Cycle problem whose complexity was left open in a 2007 paper by Arkin et al [3]. That paper is a systematic study of the complexity of the Hamiltonian Cycle problem on square, triangular, or hexagonal grid graphs, restricted to polygonal, thin, superthin, degree-bounded, or solid grid graphs. The authors solved many combinations of these problems, proving them either polynomially solvable or NP-complete, but left three combinations open. We prove two of these unsolved combinations to be NP-complete: Hamiltonian Cycle in Square Polygonal Grid Graphs and Hamiltonian Cycle in Hexagonal Thin Grid Graphs. We also consider a new restriction, where the grid graph is both thin and polygonal, and prove that the Hamiltonian Cycle problem then becomes polynomially solvable for square, triangular, and hexagonal grid graphs. Several of these results are shown by application of the TRVB results, demonstrating the usefulness of that problem.

Finally, we apply the Square Grid Graph Hamiltonian Cycle problem to close a long-standing open problem: we prove that optimally solving an $n \times n \times n$ Rubik's Cube is NP-complete. This improves the previous result that optimally solving an $n \times n \times n$ Rubik's Cube with missing stickers is NP-complete. We prove this result first for the simpler case of the Rubik's Square—an $n \times n \times 1$ generalization of the Rubik's Cube—and then proceed with a similar but more complicated proof for the Rubik's Cube case.

Thesis Supervisor: Erik Demaine
Title: Professor

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank my thesis advisor Prof. Erik Demaine for his knowledgeable and enthusiastic introduction to the world of hardness proofs and for his guidance and encouragement throughout my research. It was a pleasure working with you and learning from you. On a similar note, I would like to thank the participants of the open problem session held starting in Fall 2014 in association with MIT's class 6.890: Algorithmic Lower Bounds. I really enjoyed meeting and collaborating with you all. I would also like to acknowledge the contributions that Erik Demaine and Jayson Lynch made to this thesis by helping me with my editing.

Finally, a huge thanks to my friends and family. Your support and encouragement have put me where I am today.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

Chapter 1

Introduction

The Hamiltonian Cycle problem is one of the prototype NP-complete problems from Karp's 1972 paper [14]. Since then, many special cases of Hamiltonian Cycle have been classified as either polynomial-time solvable or NP-complete. For example, the Hamiltonian Cycle problem is known to be NP-complete in planar directed max-degree-3 graphs [19], circle graphs [8], bipartite chordal graphs [17], and square grid graphs [13, 18]. For a more comprehensive list of results about the complexity of the Hamiltonian Cycle problem restricted to various graph classes, see [9].

The many NP-hard variants of the Hamiltonian Cycle problem (restricted to various graph classes) have been the basis for many NP-hardness reductions to geometric and graph problems. In other words, these NP-hard variants of Hamiltonian Cycle form a powerful toolkit for proving NP-hardness (see [10]). In Chapter 2, we extend this toolkit by introducing a new problem, called the Tree-Residue Vertex-Breaking (TRVB) problem, which we prove NP-hard by reduction from Hamiltonian Cycle. Reducing from TRVB is often a simpler alternative to reducing from a Hamiltonian Cycle variant, so TRVB can be a powerful tool. In Chapter 3, we analyze the complexity of the Hamiltonian Cycle problem when restricted to several types of grid graphs. Some of the results in this chapter follow from the results about TRVB. Finally, in Chapter 4, we demonstrate an application for Hamiltonian Cycle. In particular, we prove that solving a given $n \times n \times n$ Rubik's Cube optimally (i.e., within a given number of moves) is NP-hard. The particular results from these chapters are described in more detail below.

1.1 Tree-Residue Vertex-Breaking

In Chapter 2, we introduce the Tree-Residue Vertex-Breaking (TRVB) problem. Given a multigraph G some of whose vertices are marked "breakable," TRVB asks whether it is possible to convert G into a tree via a sequence of applications of the vertex-breaking operation: disconnecting the edges at a degree-k breakable vertex by replacing that vertex with k degree-1 vertices (as shown in Figure 1-1). We analyze the special cases of TRVB in which the input multigraph G is restricted with any subset of the following additional constraints:

1. every breakable vertex of G must have degree from a list B of allowed degrees;
2. every unbreakable vertex of G must have degree from a list U of allowed degrees;
3. G is planar;

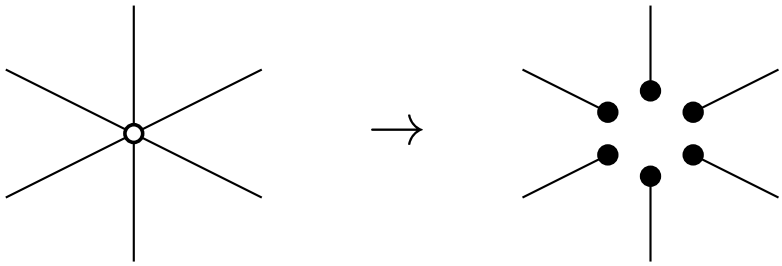


Figure 1-1: The operation of breaking a vertex. The vertex (left) is replaced by a set of degree-1 vertices with the same edges (right).

All breakable vertices have small degree ($B \subseteq \{1,2,3\}$)	**Restricted to planar simple graphs**	**All vertices have large degree ($B \cap \{1,2,3,4\} = \emptyset$ and $U \cap \{1,2,3,4,5\} = \emptyset$)**	**TRVB variant complexity**	**Section**
Yes	$*$	$*$	Polynomial Time	Section 2.3
No	No	$*$	NP-complete	Sections 2.5, 2.6, 2.7
No	Yes	No	NP-complete	Section 2.8
No	Yes	Yes	Polynomial Time (every instance is a "no" instance)	Section 2.4

Table 1.1: A summary of results from Chapter 2 (where B and U are the allowed breakable and unbreakable vertex degrees).

4. G is a simple graph (rather than a multigraph).

In Chapter 2, we fully classify these variants of TRVB into polynomial-time solvable and NP-complete. Table 1.1 summarizes our results.

1.2 How to use TRVB: Hamiltonicity in max-degree-3 square grid graphs

The TRVB problem is useful when analyzing the complexity of what could be called *single-traversal problems*: problems in which some space (i.e., a configuration graph or a grid) must be traversed in a single path or cycle subject to local constraints. Notice that variants of the Hamiltonian Cycle problem fall under the category of single-traversal problems. In this section, we show one example of using TRVB to prove hardness of a single-traversal problem. Namely, the result that Hamiltonian Cycle in max-degree-3 square grid graphs is NP-hard [18] can be reproduced with the following much simpler reduction.

The reduction is from the variant of TRVB in which the input multigraph is restricted to be planar and to have only degree-4 breakable vertices, which is shown NP-complete in

Chapter 2. Given a planar multigraph G with only degree-4 breakable vertices, we output a max-degree-3 square grid graph by appropriately placing breakable degree-4 vertex gadgets (shown in Figure 1-2) and routing edge gadgets (shown in Figure 1-3) to connect them. Each edge gadget consists of two parallel paths of edges a distance of two apart, and as shown in the figure, these paths can turn, allowing the edge to be routed as necessary (without parity constraints). Each breakable degree-4 vertex gadget joins four edge gadgets in the configuration shown. Note that, as desired, the maximum degree of any vertex in the resulting grid graph is 3.

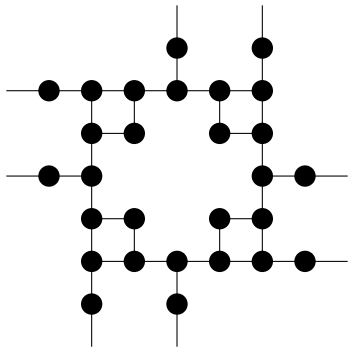


Figure 1-2: A degree-4 breakable vertex gadget.

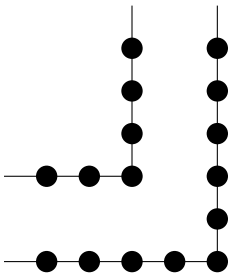


Figure 1-3: An example edge gadget consisting of two parallel paths of edges a distance of two apart.

Consider any candidate set of edges C that could be a Hamiltonian cycle in the resulting grid graph. In order for C to be a Hamiltonian cycle, C must satisfy both the local constraint that every vertex is incident to exactly two edges in C and the global constraint that C is a cycle (rather than a set of disjoint cycles). It is easy to see tha,t in order to satisfy the local constraint, every edge in every edge gadget must be in C. Similarly, there are only two possibilities within each breakable degree-4 vertex gadget which satisfy the local constraint. These possibilities are shown in Figure 1-4.

We can identify the choice of local solution at each breakable degree-4 vertex gadget with the choice of whether to break the corresponding vertex. Under this bijection, every candidate solution C satisfying local constraints corresponds with a possible residual multigraph G' formed from G by breaking vertices. The key insight is that the shape of the region R inside C is exactly the shape of G'. This is shown for an example graph-piece in Figure 1-5. The boundary of R, also known as C, is exactly one cycle if and only if R is connected and hole-free. Since the shape of region R is the same as the shape of multigraph G', this corresponds to the condition that G' is connected and acyclic, or in other words that G' is

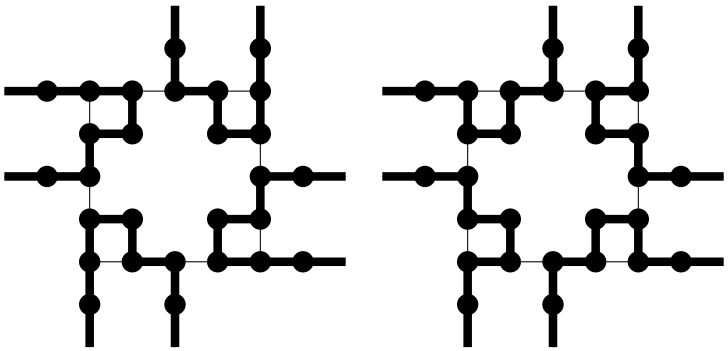


Figure 1-4: The two possible solutions to the vertex gadget from Figure 1-2 which satisfy the local constraints imposed by the Hamiltonian Cycle problem.

a tree. Thus, there exists a candidate solution C to the Hamiltonian Cycle instance that is an actual solution (satisfying both the global and local constraints) if and only if G is a "yes" instance of TRVB. Therefore, Hamiltonian Cycle in max-degree-3 square grid graphs is NP-hard.

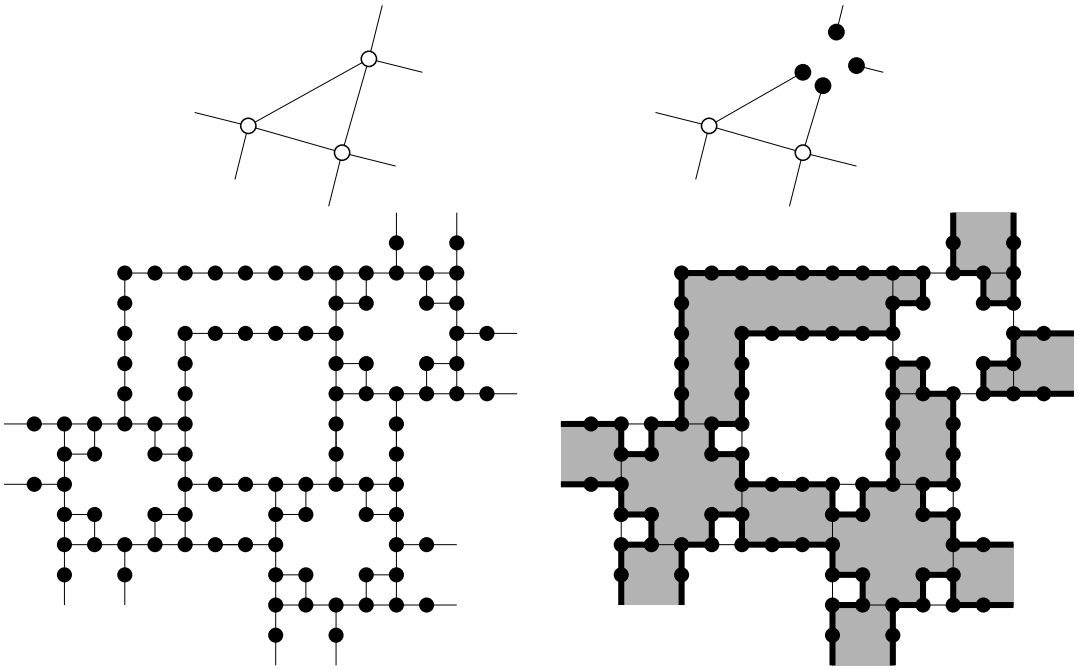


Figure 1-5: Given a multigraph including the piece shown in the top left, the output grid graph might include the section shown in the bottom left (depending on graph layout). If the top vertex in this piece of the multigraph is broken, resulting in the piece of multigraph G' shown in the top right, then the resulting candidate solution C (shown in bold) in the bottom right contains region R (shown in grey) whose shape resembles the shape of G'.

This same proof idea can apply to any single-traversal problem as long as we demonstrate both an edge gadget and a breakable degree-k vertex gadget for some $k \geq 4$. In order for the proof to go through, the edge gadget must contain two parallel paths, both of which must be traversed due to the local constraints of the single-traversal problem. In addition,

the vertex gadget must have exactly two possible solutions satisfying the local constraints of the problem: one solution will disconnect the regions inside all the adjoining edge gadgets, while the other will connect these regions inside the vertex gadget.

1.3 Hamiltonian Cycle in grid graphs

An important NP-complete special case of the Hamiltonian Cycle problem is its restriction to *(square) grid graphs* [13], where vertices lie on the 2D integer square grid and edges connect all unit-distance vertex pairs. Hamiltonian Cycle in grid graphs has been used as the source problem for NP-hardness reductions to many geometric and planar-graph problems, such as Euclidean TSP [13], Euclidean degree-bounded minimum spanning tree [18], 2D platform games with item collection and time limits [12], the Slither Link puzzle [26], the Hashiwokakero puzzle [1], lawn mowing and milling (e.g., 3D printing) [4], and minimum-turn milling [2]; see [10].

Given all these applications, it is natural to wonder how special we can make the grid graphs, and whether we can change the grid to triangular or hexagonal, and still keep Hamiltonian Cycle NP-complete. Two notable examples are NP-completeness in maximum-degree-3 square grid graphs [18] and a polynomial-time algorithm for *solid* square grid graphs [22]. In 2007, Arkin et al. [3] initiated a systematic study of the complexity of Hamiltonian Cycle in square, triangular, or hexagonal grid graphs, restricted to several special cases: polygonal, thin, superthin, degree-bounded, or solid grid graphs. See [3] or Section 3.2 for definitions. Table 1.2 (nonbold) summarizes the many results they obtained, including several NP-completeness results and a few polynomial-time algorithms.

Arkin et al. [3] left unsolved three of the combinations between grid shape and special property: Hamiltonian Cycle in Square Polygonal Grid Graphs, Hamiltonian Cycle in Hexagonal Thin Grid Graphs, and Hamiltonian Cycle in Hexagonal Solid Grid Graphs. We prove that the first two of these, Hamiltonian Cycle in Square Polygonal Grid Graphs and Hamiltonian Cycle in Hexagonal Thin Grid Graphs, are NP-complete. In addition, we consider another case not considered in that paper, namely, *thin polygonal* grid graphs (the fusion of two special cases). We show that Hamiltonian Cycle becomes polynomially solvable in this case, for all three shapes of grid graph. Table 1.2 (bold) summarizes our new results. Among these results, the ones pertaining to hexagonal grid graphs are obtained by reductions to and from variants of TRVB.

Grid	Triangular	Square	Hexagonal
General	NP-complete	NP-complete	NP-complete
Degree-bounded	$\deg \leq 3$ NP-complete	$\deg \leq 3$ NP-complete	$\deg \leq 2$ Polynomial
Thin	NP-complete	NP-complete	**NP-complete**
Superthin	NP-complete	Polynomial	Polynomial
Polygonal	Polynomial	**NP-complete**	NP-complete
Solid	Polynomial	Polynomial	Open
Thin Polygonal	**Polynomial**	**Polynomial**	**Polynomial**

Table 1.2: Complexity of Hamiltonian Cycle in grid graph variants; bold entries correspond to new results in Chapter 3 (see [3] or Section 3.2 for definitions).

1.4 Solving the Rubik's Cube optimally

The Rubik's Cube is an iconic puzzle in which the goal is to rearrange the stickers on the outside of a $3 \times 3 \times 3$ cube so as to make each face monochromatic by rotating $1 \times 3 \times 3$ (or $3 \times 1 \times 3$ or $3 \times 3 \times 1$) slices. The $3 \times 3 \times 3$ Rubik's Cube can be generalized to an $n \times n \times n$ cube in which a single move is a rotation of a $1 \times n \times n$ slice. We can also consider the generalization to a $n \times n \times 1$ figure. In this simpler puzzle, called the Rubik's Square, the allowed moves are flips of $n \times 1 \times 1$ rows or $1 \times n \times 1$ columns. These two generalizations were introduced in [11].

The question of whether optimally solving the $n \times n \times n$ Rubik's Cube (i.e., solving it in the fewest moves) is NP-complete has been repeatedly posed as far back as 1984 [6, 20, 23] and has until now remained open [15]. A previous result (see [11]) showed that optimally solving the Rubik's Square is NP-complete with "wildcard" stickers which are allowed on any face of the solved puzzle. In Chapter 4, we first prove that optimally solving the Rubik's Square is NP-hard by reduction from a variant of Hamiltonian Cycle, and then proceed to apply the same ideas to a more complicated proof of NP-hardness for optimally solving the Rubik's Cube.

Chapter 2

Tree-Residue Vertex-Breaking

2.1 Introduction

The multigraph operation of *breaking* vertex v in undirected multigraph G results in a new multigraph G' by removing v, adding a number of new vertices equal to the degree of v in G, and connecting these new vertices to the neighbors of v in G in a one-to-one manner, as shown in Figure 2-1.

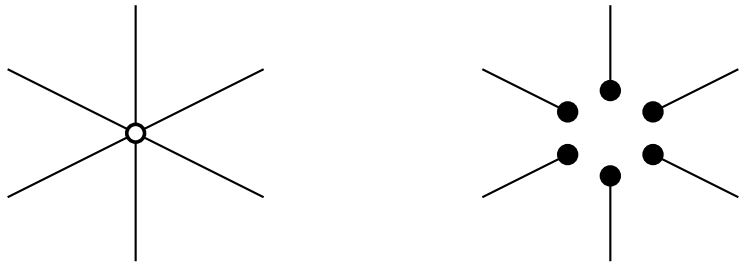


Figure 2-1: The operation of breaking a vertex. The vertex (left) is replaced by a set of degree-1 vertices with the same edges (right).

Using this definition, we pose the following computational problem:

Problem 2.1. *The* Tree-Residue Vertex-Breaking Problem (TRVB) *takes as input a multigraph G whose vertices are partitioned into two sets V_B and V_U (called the* breakable *and* unbreakable *vertices respectively), and asks to decide whether there exists a set $S \subseteq V_B$ such that after breaking every vertex of S in G, the resulting multigraph is a tree.*

In order to avoid trivial cases, we consider only input graphs that have no degree-0 vertices.

In this chapter, we analyze this problem as well as several variants (special cases) where G is restricted with any subset of the following additional constraints:

1. every breakable vertex of G must have degree from a list B of allowed degrees;
2. every unbreakable vertex of G must have degree from a list U of allowed degrees;
3. G is planar;
4. G is a simple graph (rather than a multigraph).

All breakable vertices have small degree ($B \subseteq \{1,2,3\}$)	**Restricted to planar simple graphs**	**All vertices have large degree** ($B \cap \{1,2,3,4\} = \emptyset$ **and** $U \cap \{1,2,3,4,5\} = \emptyset$)	**TRVB variant complexity**	**Section**
Yes	$*$	$*$	Polynomial Time	Section 2.3
No	No	$*$	NP-complete	Sections 2.5, 2.6, 2.7
No	Yes	No	NP-complete	Section 2.8
No	Yes	Yes	Polynomial Time (every instance is a "no" instance)	Section 2.4

Table 2.1: A summary of this chapter's results (where B and U are the allowed breakable and unbreakable vertex degrees).

Modifying TRVB to include these constraints makes it easier to reduce from the TRVB problem to some other. For example, having a restricted list of possible breakable vertex degrees B allows a reduction to include gadgets only for simulating breakable vertices of those degrees, whereas without that constraint, the reduction would have to support simulation of breakable vertices of any degree.

We prove the following results (summarized in Table 2.1), which together fully classify the variants of TRVB into polynomial-time solvable and NP-complete problems:

1. Every TRVB variant whose breakable vertices are only allowed to have degrees of at most 3 is solvable in polynomial time.

2. Every planar graph TRVB variant whose breakable vertices are only allowed to have degrees of at least 6 and whose unbreakable vertices are only allowed to have degrees of at least 5 is solvable in polynomial time (and in fact the correct output is always "no").

3. In all other cases, the TRVB variant is NP-complete. In particular, the TRVB variant is NP-complete if the variant allows breakable vertices of some degree $k > 3$ and in the planar graph case also allows either breakable vertices of degree $b < 6$ or unbreakable vertices of degree $u < 5$.

In Section 2.2, we begin by formally defining these variants of TRVB. We also prove membership in NP for all the variants and provide the obvious reductions between them.

In Section 2.3, we prove that, if every breakable vertex is restricted to have degree at most 3, then the TRVB variant in question is solvable in polynomial time. Next, in Section 2.4, we prove that if the multigraph is restricted to be a planar graph, the breakable vertices are restricted to have degree at least 6, and the unbreakable vertices are restricted to have degree at least 5, then it is impossible to break a set of breakable vertices and get a tree. Therefore, in these cases, the TRVB variant is solvable in polynomial time.

Next we prove our NP-hardness results. In Section 2.5, we reduce from an NP-hard problem to show that Planar TRVB with only degree-k breakable vertices and unbreakable degree-4 vertices is NP-hard for any $k \geq 4$. All the other hardness results in this chapter are derived directly or indirectly from this one. In Section 2.6, we prove the NP-completeness of the variants of TRVB and of Planar TRVB in which breakable vertices of some degree $k \geq 4$ are allowed. Similarly, we show in Section 2.7 that Graph TRVB is also NP-complete in the presence of breakable vertices of degree $k \geq 4$. Finally, in Section 2.8, we show that Planar Graph TRVB is NP-complete provided (1) breakable vertices of some degree $k \geq 4$ are allowed and (2) either breakable vertices of degree $b \leq 5$ or unbreakable vertices of degree $u \leq 4$ are allowed.

All together, these results completely classify the variants of TRVB into NP-complete and polynomially solvable.

The research in this chapter is joint work with Erik Demaine. We would also like to thank Zachary Abel and Jayson Lynch for contributing to the research.

2.2 Problem variants

2.2.1 Problem variant definitions

In this section, we will formally define the variants of TRVB under consideration.

To begin, suppose B and U are both sets of positive integers. Then we can constrain the breakable vertices of the input to have degrees in B and constrain the unbreakable vertices of the input to have degrees in U. The resulting constrained version of the problem is defined below:

Definition 2.2. *The* (B, U)-variant of the TRVB problem*, denoted* (B, U)*-TRVB, is the special case of TRVB where the input multigraph is restricted so that every breakable vertex in* G *has degree in* B *and every unbreakable vertex in* G *has degree in* U*.*

Throughout this chapter we consider only sets B and U for which membership can be computed in pseudopolynomial time (i.e., membership of n in B or U can be computed in time polynomial in n). As a result, verifying that the vertex degrees of a given multigraph are allowed can be done in polynomial time. This means that the classification of a particular (B, U)-variant of the TRVB problem into polynomially solvable or NP-complete is a statement about the hardness of checking TRVB (while constrained by the other conditions) rather than a statement about the hardness of checking membership in B or U for the degrees in the multigraph. In fact, all the results in this chapter will also apply even in the cases that membership in B or U cannot be computed in pseudo-polynomial time if we consider the promise problems in which the given multigraph's vertex degrees are guaranteed to comply with the sets B and U.

We can also define three further variants of the problem depending on whether G is constrained to be planar, a graph, or both: the *Planar* (B, U)*-variant of the TRVB problem* (denoted Planar (B, U)-TRVB), the *Graph* (B, U)*-variant of the TRVB* (denoted Graph (B, U)-TRVB), and the *Planar Graph* (B, U)*-variant of the TRVB problem* (denoted Planar Graph (B, U)-TRVB).

Since both being planar and being a graph are properties of a multigraph that can be verified in polynomial time, the classification of these variants into polynomially solvable or NP-complete is again a statement about the hardness of TRVB.

2.2.2 Diagram conventions

Throughout this chapter, when drawing diagrams, we will use filled circles to represent unbreakable vertices and unfilled circles to represent breakable vertices. See Figure 2-2.

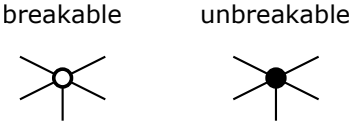


Figure 2-2: An example diagram, showing the depictions of vertex types used in this chapter.

2.2.3 Trivial reductions

As mentioned above, except for the constraint that the TRVB problem outputs "yes" on the given input, every other constraint in the definition of each of the above variants can be tested in polynomial time. Therefore, if for some two variants X and Y the non-TRVB conditions of X are strictly stronger (more constraining) than the non-TRVB conditions of Y, then we can reduce from X to Y in polynomial time. In particular, we can convert an input G for variant X into an input G' for Y as follows:

First test all the non-TRVB conditions of variant X on the input G. If any condition is not satisfied, then X rejects G, so output any G' rejected by Y. If all the non-TRVB conditions of variant X are satisfied, then by assumption all the non-TRVB conditions of variant Y on input G are also satisfied. Therefore G is a "yes" instance of both X and Y if and only if G is a "yes" instance of TRVB. Therefore X and Y have the same answer on G, so outputting $G' = G$ completes the reduction.

Using the above reduction scheme, we conclude that:

Lemma 2.3. *For any (B, U), there are reductions*

- *from Planar (B, U)-TRVB to (B, U)-TRVB,*
- *from Graph (B, U)-TRVB to (B, U)-TRVB,*
- *from Planar Graph (B, U)-TRVB to Planar (B, U)-TRVB, and*
- *from Planar Graph (B, U)-TRVB to Graph (B, U)-TRVB.*

For any (B, U) and (B', U') with $B \subseteq B'$ and $U \subseteq U'$, there are reductions

- *from (B, U)-TRVB to (B', U')-TRVB,*
- *from Planar (B, U)-TRVB to Planar (B', U')-TRVB,*
- *from Graph (B, U)-TRVB to Graph (B', U')-TRVB, and*
- *from Planar Graph (B, U)-TRVB to Planar Graph (B', U')-TRVB.*

2.2.4 Membership in NP

Theorem 2.4. *The TRVB problem is in NP.*

Proof. We describe a nondeterministic algorithm to solve TRVB: First nondeterministically guess a set of breakable vertices in G. Break that set of vertices and accept if and only if the resulting multigraph is a tree.

This algorithm accepts an input G on at least one nondeterministic branch if and only if it is possible to break some of the breakable vertices so that the residual multigraph is a tree. In other words, this algorithm solves TRVB. Furthermore, the algorithm runs in polynomial time since both breaking vertices and checking whether a multigraph is a tree are polynomial-time operations. As desired, TRVB is in NP. □

Another name for TRVB is $(\mathbb{N}, \mathbb{N})$-TRVB, so we can apply the reductions from Lemma 2.3 to conclude that:

Corollary 2.5. *For any (B, U), the (B, U)-TRVB, Planar (B, U)-TRVB, Graph (B, U)-TRVB, and Planar Graph (B, U)-TRVB are in NP.*

2.3 TRVB is polynomial-time solvable without high-degree breakable vertices

The overall goal of this section is to show that the variants of TRVB without breakable vertices of degree $k > 3$ are polynomial-time solvable. In particular, (B, U)-TRVB is polynomial-time solvable if $B \subseteq \{1, 2, 3\}$ (and thus so are Planar (B, U)-TRVB, Graph (B, U)-TRVB, and Planar Graph (B, U)-TRVB).

We begin by proving that $(\{3\}, \emptyset)$-TRVB is polynomial-time solvable in Section 2.3.1 as a warmup. We prove this by reducing $(\{3\}, \emptyset)$-TRVB to the Cubic Nonseparating Independent Set problem, which was shown to be polynomial-time solvable in [21]. This warmup is useful because it introduces the Cubic Nonseparating Independent Set problem, points out the connection between this problem and Tree-Residue Vertex-Breaking, and proves a few useful lemmas; the actual proof that $(\{3\}, \emptyset)$-TRVB is polynomial-time solvable (the main part of Section 2.3.1), however, is purely a warmup and can be safely skipped.

Next, in Section 2.3.2, we introduce the Nonseperating Independent Tag Set problem and prove that it is polynomial-time solvable. The Nonseperating Independent Tag Set problem is a generalization of the Cubic Nonseparating Independent Set problem, and the proof that Nonseperating Independent Tag Set is polynomial-time solvable follows the same approach as was used in [21] for the Cubic Nonseparating Independent Set problem.

Finally, in Section 2.3.3, we reduce from $(\{1, 2, 3\}, \mathbb{N})$-TRVB to Nonseperating Independent Tag Set, thereby concluding via Lemma 2.3 that (B, U)-TRVB, Planar (B, U)-TRVB, Graph (B, U)-TRVB, and Planar Graph (B, U)-TRVB with $B \subseteq \{1, 2, 3\}$ can be solved in polynomial time.

2.3.1 $(\{3\}, \emptyset)$-TRVB is polynomial-time solvable

We show that $(\{3\}, \emptyset)$-TRVB (the Tree-Residue Vertex-Breaking problem with all vertices breakable and degree-3) is polynomial-time solvable by reduction to the Cubic Nonseparating Independent Set problem, which was shown to be solvable in polynomial time in [21]. We define the problem and related terms below:

Definition 2.6. *A set of vertices X of multigraph G is a* separating set *if the number of connected components of $G - X$ is more than that of G.*

Definition 2.7. *A set of vertices S in a multigraph is an* independent set *if no two vertices in the set are adjacent and is a* nonseparating set *if every $X \subseteq S$ is not a separating set.*

Problem 2.8. *The* Cubic Nonseparating Independent Set problem *asks, for a given cubic multigraph G and a given number s, whether these exists a nonseparating independent set S in G with $|S| = s$.*

The following lemma is an important connection between vertex breaking and nonseperating independent sets.

Lemma 2.9. *Suppose that G is a connected multigraph and S is a subset of the vertices in G. Let G' be the graph formed by breaking all the vertices of S in G. Then G' is connected if and only if S is a nonseparating independent set.*

Proof. First suppose that G' is connected.

Let A be any subset of S. Let G'_A be the multigraph formed by breaking all the vertices of A in G. We know that G' can be formed from G'_A by breaking the vertices in $S - A$. Since the reverse operation of vertex breaking is a merging of several degree-1 vertices, we conclude that we can construct G'_A from G' by merging vertices. Then since merging vertices only increases connectivity and G' is connected, we can conclude that G'_A is connected as well. Let G''_A be the multigraph formed by removing the vertices of A from G. Vertex breaking is equivalent to vertex removal followed by an addition of a new degree-1 vertex (also adding that vertex's sole edge). Therefore, it is possible to transform G''_A into G'_A by inserting new degree-1 vertices. Alternatively, this means we can transform G'_A into G''_A by removing degree-1 vertices. Removing a degree-1 vertex never disconnects a multigraph, so since G'_A is connected, we conclude that G''_A is connected. We have shown that removing the vertices of A from G (which yields G''_A) does not disconnect the multigraph. Therefore A is not a separating set of G. Since this is true for every $A \subseteq S$, we conclude that by definition, S is a nonseparating set.

Furthermore, no two vertices in S can be adjacent because that would lead to a separation of the edge between them from the rest of the multigraph (in G'), so S is an independent set. Thus, we have shown that if G' is connected, then S is a nonseparating independent set.

On the other hand, suppose S is a nonseparating independent set. Let G'' be the multigraph formed by removing the vertices of S from G.

Let T be the set of vertices in G but not S. The set of vertices of G'' is exactly T. The set of vertices of G' contains T. The only vertices in G' that are not in T are the degree-1 vertices that are added to G during the vertex breaking operation. Note that by construction, if t_1 and t_2 are in T, then the edge (t_1, t_2) is either in all three graphs G, G', and G'', or in none of them. Since S is nonseparating, G'' has the same number of components as G: precisely one. Then since T is the vertex set of G'' and G'' is connected, there is a path in G'' between any two vertices in T (using only vertices in T). Every such path must also exist in G', and so all of T is in the same connected component of G'.

Consider any vertex v' in G' that is not in T. This vertex v' was added to G' while breaking some vertex $v \in S$. At the time of the breaking operation, some neighbor u of v instead became the sole neighbor of v'. Since $v \in S$ and S is an independent set in G, we know that u, which is a neighbor of v in G cannot also be in S. Thus $u \notin S$ and therefore $u \in T$. Then the edge (u, v') in G' connects v' to the connected component of G' containing

all of T. Since this applies to every $v' \notin T$, we can conclude that all of G' is in the same connected component. In other words, we have shown that G' is connected whenever S is a nonseparating independent set.

As desired, we have shown both directions. We conclude that G' is connected if and only if S is a nonseparating independent set. □

Next, we prove the main theorem of this section, which as previously mentioned will serve as a warmup for the further results.

Theorem 2.10. $(\{3\}, \emptyset)$*-TRVB is polynomial-time solvable.*

Proof. Suppose that G is a multigraph in which every vertex is breakable and degree-3. Then if G contains n vertices, we know that G must contain $\frac{3n}{2}$ edges. Breaking a degree-3 vertex leaves the number of edges unchanged and increases the number of vertices by 2. Therefore if a solution to the Tree-Residue Vertex-Breaking problem on G breaks b vertices, the final multigraph will have $n + 2b$ vertices and $\frac{3n}{2}$ edges.

In order for a solution to the Tree-Residue Vertex-Breaking problem on G to be valid, the resulting multigraph must be a tree. Therefor the number of vertices is one more than the number of edges, or in other words $(n + 2b) = (\frac{3n}{2}) + 1$. Thus $b = \frac{n+2}{4}$.

We see that any valid solution to the Tree-Residue Vertex-Breaking problem on G will break exactly $\frac{n+2}{4}$ vertices.

In combination with Lemma 2.9, this tells us that any valid solution to the Tree-Residue Vertex-Breaking problem on G breaks a set of vertices S where S is a nonseparating independent set and $|S| = \frac{n+2}{4}$.

On the other hand, suppose S is a nonseparating independent set of size $\frac{n+2}{4}$. By Lemma 2.9, the multigraph G' formed by breaking the vertices of S in G is connected. Since every vertex has degree 3, the breaking of each element of S in G increases the number of vertices by 2 and leaves the number of edges the same. Thus the number of vertices in G' is $n + 2(\frac{n+2}{4}) = \frac{3n}{2} + 1$ and the number of edges in G' is $\frac{3n}{2}$. Since G' is a connected multigraph which has one more vertex than it has edges, we can conclude that G' is a tree. In other words, breaking the vertices of S is a valid solution to the Tree-Residue Vertex-Breaking problem on G.

We conclude that the nonseparating independent sets of G of size $\frac{n+2}{4}$ are exactly the sets of broken vertices from the valid solutions of the Tree-Residue Vertex-Breaking problem on G. Since the Cubic Nonseparating Independent Set problem can be solved in polynomial time, this allows us to also solve $(\{3\}, \emptyset)$-TRVB in polynomial time. □

2.3.2 Nonseparating Independent Tag Set is polynomial-time solvable

The purpose of this section is to introduce the following problem and prove that it is polynomial-time solvable:

Problem 2.11. *Define a* tagged graph *to be a graph together with a partial labeling of the vertices such that (1) every vertex is labeled with either zero or one tags and (2) every tag either labels one degree-*3 *vertex or labels two degree-*2 *vertices.*

The Nonseparating Independent Tag Set *problem asks, for a given tagged graph G and a given number s, whether there exists a set S of tags such that $|S| = s$ and such that S', the set of vertices in G which are labeled with tags in S, is a nonseparating independent set.*

Notice that on inputs that are cubic graphs whose vertices are each uniquely labeled, a solution to the Nonseperating Independent Tag Set problem corresponds exactly to a nonseperating independent set in that graph. In other words, the Nonseperating Independent Tag Set problem is a generalization of the Cubic Nonseperating Independent Set problem. The Cubic Nonseparating Independent Set problem was shown to be polynomial-time solvable in [21] via a reduction to the Matching problem for 2-polymatroids (introduced later in this section), which itself was shown to be polynomial-time solvable in [16]. We will expand upon this result, providing a very similar reduction from the more general Nonseperating Independent Tag Set problem to the Matching problem for 2-polymatroids.

To define the Matching problem for 2-polymatroids, we need to introduce several new terms.

Definition 2.12. *A* polymatroid *is a pair (X, f) where X is a finite set and f is a function defined on subsets of X such that*

1. *$f(\emptyset) = 0$,*
2. *$f(A_1) \leq f(A_2)$ if $A_1 \subseteq A_2 \subseteq X$, and*
3. *$f(A_1 \cup A_2) + f(A_1 \cap A_2) \leq f(A_1) + f(A_2)$ for any $A_1, A_2 \subseteq X$.*

A k-polymatroid *is a polymatroid (X, f) satisfying the additional condition that $f(\{x\}) \leq k$ for all $x \in X$. A* matroid *is a 1-polymatroid.*

Definition 2.13. *A* matching *of a 2-polymatroid (X, f) is a set $Y \subseteq X$ such that $f(Y) = 2|Y|$.*

At this point, we are almost ready to define the Matching problem for 2-polymatroids. The issue is that polymatroids in general are computationally intractible to deal with; just the description of a polymatroid (X, f) has length exponential in $|X|$. This is not the case for a restricted subset of all polymatroids: "linearly represented" polymatroids.

"Linearly represented" polymatroids are defined in terms of linearly represented matroids below, where we treat linearly represented matroids as a black box whose definition will not be given in this chapter. For the definition of and other information on linearly represented matroids, see [24].

Definition 2.14. *A* linearly represented polymatroid *is a polymatroid with a "linear representation." A* linear representation *of polymatroid (X, f) is a linearly represented matroid (E, r) and an assignment of subsets E_x to elements x of X such that, for any $Y \subseteq X$,*

$$f(Y) = r\left(\bigcup_{x \in Y} E_x\right).$$

With that done, we can finally define the Matching problem for 2-polymatroids:

Problem 2.15. *The* Matching problem for linearly represented 2-polymatroids *asks, for a given linear representation of 2-polymatroid (X, f) and a given number s, whether it is possible to find a matching of (X, f) of size at least s.*

We wish to reduce from the Nonseperating Independent Tag Set problem to the Matching problem for 2-polymatroids, so we need some way to connect graphs and polymatroids. This is done through the following definition.

Definition 2.16. *Define the* circuit rank $\mu(G)$ *of graph* G *to be* $m - n + c$ *where* m*,* n*, and* c *are the edge, vertex, and connected-component counts in* G*.*

Suppose G *is a graph. If* V *is the set of vertices of* G*, then we define* $P(G)$ *to equal* (V, f) *where* f *is a function defined on subsets of* V *such that for* $V' \subseteq V$ *we define* $f(V') = \mu(G) - \mu(G - V')$*.*

Suppose next that G *is a tagged graph and* T *is the set of tags in* G*. Let* $(V, f) = P(G)$*, and for* $t \in T$ *let* $e(t)$ *be the set of vertices in* V *labeled with tag* t*. Then we define* $\hat{P}(G) = (T, f')$ *where* f' *is a function defined on subsets of* T *such that, for any* $T' \subseteq T$*,*

$$f'(T') = f\left(\bigcup_{t \in T'} e(t)\right).$$

Then the reduction we are looking for takes as input a tagged graph G and a number s (an instance of Nonseperating Independent Tag Set) and outputs a linear representation of $\hat{P}(G)$ and s (an instance of the Matching problem for 2-polymatroids).

Later in this section, we will prove two major lemmas, stated below, which will allow us to show that this reduction is both correct and computable in polynomial time:

Lemma 2.17. *For any tagged graph* G*,* $\hat{P}(G)$ *is a linearly representable* 2*-polymatroid whose linear representation can be identified in polynomial time.*

Lemma 2.18. T' *is a matching of* $\hat{P}(G) = (T, f')$ *if and only if the set of vertices* S *labeled with tags in* T' *is a nonseparating independent set.*

Theorem 2.19. *The Nonseperating Independent Tag Set problem is polynomial-time solvable.*

Proof. We can immediately conclude from Lemma 2.17 that the outputs of the reduction (a linear representation of $\hat{P}(G)$ and s) can be computed from the inputs (G and s) in polynomial time. Lemma 2.17 also tells us that $\hat{P}(G)$ is a 2-polymatroid, which makes the output of the reduction a valid instance of the Matching problem for 2-polymatroids. In addition, we know from Lemma 2.18 that the sets of tags for which the corresponding sets of vertices are nonseparating independent sets are exactly the matchings in $\hat{P}(G)$. Thus, $\hat{P}(G)$ has a matching of size s if and only if G has a set of tags of size s whose corresponding set of vertices is a nonseparating independent set. In other words, there exists a solution to the Nonseperating Independent Tag Set instance specified by G and s if and only if there exists a solution to the Matching for 2-polymatroids instance specified by a linear representation of $\hat{P}(G)$ and s.

Thus, the reduction is correct and can be run in polynomial time, so we can conclude that as desired, the Nonseperating Independent Tag Set problem is polynomial-time solvable. □

All that is left is to prove the two lemmas above.

We begin with Lemma 2.17. It was shown in [21] that for any graph G, $P(G)$ is a polymatroid. The paper then proves that a linear representation for $P(G)$ can be found in polynomial time. In order to make use of this fact, we will prove the following lemma:

Lemma 2.20. *Suppose* (X, f) *is a polymatroid,* (L, h) *is a linearly represented polymatroid, and each element* x *of* X *is assigned a subset* $L_x \subset L$ *such that for any* $Y \subseteq X$*,*

$$f(Y) = h\left(\bigcup_{x \in Y} L_x\right).$$

Then a linear representation for (X, f) can be computed from a linear representation of (L, h) in polynomial time.

Proof. Let (E, r) and $\{E_l\}_{l \in L}$ be a linear representation of (L, h).

For each $x \in X$ define E'_x to be $\bigcup_{l \in L_x} E_l$. Then (E, r) and $\{E'_x\}_{x \in X}$ is the linear representation of (X, f) whose existence is claimed by this lemma. We can certainly compute (E, r) and $\{E'_x\}_{x \in X}$ in polynomial time from (E, r) and $\{E_l\}_{l \in L}$, so provided this really is a linear representation, we will have shown our desired result.

In order for (E, r) and $\{E_l\}_{l \in L}$ to be the linear representation of (L, h), the object (E, r) must be a linearly representable matroid. And since $E_l \subseteq E$ for all $l \in L$, we have that $E'_x = \bigcup_{l \in L_x} E_l \subseteq E$ for all $x \in X$. Thus in order to show that (E, r) and $\{E'_x\}_{x \in X}$ is a linear representation of (X, f), all that is left is to show that for any set $Y \subseteq X$, it is the case that

$$f(Y) = r\left(\bigcup_{x \in Y} E'_x\right).$$

Define $L_Y = \bigcup_{x \in Y} L_x$. We can apply the definition of E'_x to see that

$$r\left(\bigcup_{x \in Y} E'_x\right) = r\left(\bigcup_{x \in Y} \bigcup_{l \in L_x} E_l\right).$$

Then substituting in the definition of L_y, we see that

$$r\left(\bigcup_{x \in Y} \bigcup_{l \in L_x} E_l\right) = r\left(\bigcup_{l \in L_Y} E_l\right).$$

Next, we apply the definition of a linear representation to linear representation (E, r) and $\{E_l\}_{l \in L}$ of (L, h) to get

$$r\left(\bigcup_{l \in L_Y} E_l\right) = h(L_y).$$

But by definition of L_y,

$$h(L_y) = h\left(\bigcup_{x \in Y} L_x\right).$$

Finally, applying the assumption given in this lemma, we see that

$$h\left(\bigcup_{x \in Y} L_x\right) = f(Y).$$

Putting this all together, we have that $f(Y) = r\left(\bigcup_{x \in Y} E'_x\right)$ as desired. □

With this lemma in place, we procede to show that $\hat{P}(G)$ is a linearly representable polymatroid whose linear representation can be identified in polynomial time.

Lemma 2.21. *For any tagged graph G, $\hat{P}(G)$ is a linearly representable polymatroid whose linear representation can be identified in polynomial time.*

Proof. Suppose that $\hat{P}(G) = (T, f')$ and $P(G) = (V, f)$.

We first verify that (T, f') is a polymatroid.

Consider $f'(\emptyset)$. This value is defined to be $f\left(\bigcup_{t\in\emptyset} e(t)\right) = f(\emptyset)$, so since $P(G) = (V, f)$ is a polymatroid, we have that $f(\emptyset) = 0$. Thus $f'(\emptyset) = 0$.

Consider any $A_1 \subseteq A_2 \subseteq T$. We know that $f'(A_1) = f\left(\bigcup_{t\in A_1} e(t)\right)$ and that $f'(A_2) = f\left(\bigcup_{t\in A_2} e(t)\right)$ Then if we let $B_1 = \bigcup_{t\in A_1} e(t)$ and let $B_2 = \bigcup_{t\in A_2} e(t)$, we see that $B_1 \subseteq B_2 \subseteq V$, and therefore (since $P(G) = (V, f)$ is a polymatroid) $f(B_1) \leq f(B_2)$. But then $f'(A_1) = f(B_1) \leq f(B_2) = f'(A_2)$ so $f'(A_1) \leq f'(A_2)$.

Consider any $A_1, A_2 \subseteq T$. Define $B_1 = \bigcup_{t\in A_1} e(t)$ and $B_2 = \bigcup_{t\in A_2} e(t)$. Then we have

$$f'(A_1 \cup A_2) = f\left(\bigcup_{t\in A_1\cup A_2} e(t)\right) = f\left(\bigcup_{t\in A_1} e(t) \cup \bigcup_{t\in A_2} e(t)\right) = f(B_1 \cup B_2),$$

$$f'(A_1) = f\left(\bigcup_{t\in A_1} e(t)\right) = f(B_1),$$

and

$$f'(A_2) = f\left(\bigcup_{t\in A_2} e(t)\right) = f(B_2).$$

Using the fact that the sets $e(t)$ are disjoint for different t, we can also show that

$$f'(A_1 \cap A_2) = f\left(\bigcup_{t\in A_1\cap A_2} e(t)\right) = f\left(\bigcup_{t\in A_1} e(t) \cap \bigcup_{t\in A_2} e(t)\right) = f(B_1 \cap B_2).$$

Then the statement $f(B_1 \cup B_2) + f(B_1 \cap B_2) \leq f(B_1) + f(B_2)$ (true because $P(G) = (V, f)$ is a polymatroid) can be rewritten as $f'(A_1 \cup A_2) + f'(A_1 \cap A_2) \leq f'(A_1) + f'(A_2)$

As we see, (T, f') satisfies all the conditions in the definition of a polymatroid and therefore is one.

Next, notice that if $T' \subseteq T$, then $f'(T') = f\left(\bigcup_{t\in T'} e(t)\right)$. Because of this property, we can apply Lemma 2.20: it is possible to compute a linear representation for $\hat{P}(G) = (T, f')$ from a linear representation of $P(G) = (V, f)$. Since a linear representation for $P(G)$ can be computed in polynomial time, this means that a linear representation for $\hat{P}(G)$ can be computed in polynomial time. □

We have shown that, for a tagged graph G, the object $\hat{P}(G)$ is a polymatroid whose linear representation can be computed in polynomial time. Consider Lemma 2.17 reproduced below. All that we have left to show in order to prove Lemma 2.17 is that $\hat{P}(G)$ is actually a 2-polymatroid.

Lemma 2.17. *For any tagged graph G, $\hat{P}(G)$ is a linearly representable 2-polymatroid whose linear representation can be identified in polynomial time.*

Proof. Suppose that $\hat{P}(G) = (T, f')$ and $P(G) = (V, f)$. We must show that, for any $t \in T$, $f'(\{t\}) \leq 2$.

There are two cases. Either t is a tag labeling two degree-2 nodes or t is a tag labeling one degree-3 node. If t is a tag labeling two degree-2 nodes, then $e(t) = \{v_1, v_2\}$ where $v_1, v_2 \in V$ have degree 2. Then $f'(\{t\}) = f(\{v_1, v_2\}) \leq f(\{v_1\}) + f(\{v_2\})$. If t is a tag labeling one degree-3 node, then $e(t) = \{v\}$ where $v \in V$ has degree 3. Then $f'(\{t\}) = f(\{v\})$.

For any vertex v, $f(\{v\}) = \mu(G) - \mu(G - \{v\})$. If we define Δm, Δn, and Δc to be the increases in edge, vertex, and connected component counts due to adding vertex v into graph $G - \{v\}$, then $f(\{v\}) = \Delta m - \Delta n + \Delta c$. Note that Δm is the degree of v in G, $\Delta n = 1$ and $\Delta c \leq 0$ (since adding a vertex that is connected to something else cannot increase the number of connected components). Therefore we see that for any vertex v, $f(\{v\})$ is bounded above by the degree of v minus one.

Thus in the first case $f'(\{t\}) = f(\{v_1, v_\}) \leq f(\{v_1\}) + f(\{v_2\}) \leq (2-1) + (2-1) = 2$ and in the second case $f'(\{t\}) = f(\{v\}) \leq (3-1) = 2$. In both cases, we have concluded that $f'(\{t\}) \leq 2$, so we can conclude that $\hat{P}(G)$ is a 2-polymatroid. □

With that done, we can proceed to the last remaining part of the proof: the proof of Lemma 2.18. Consider first the following lemma connecting $\hat{P}(G)$ and G:

Lemma 2.22. *Define $d_G(v)$ to be the degree of vertex v in graph G. Suppose for some graph G we have $P(G) = (V, f)$. Then for any set $X \subseteq V$, $f(X) = \sum_{v \in X}(d_G(v) - 1)$ if and only if X is a nonseparating independent set in G.*

Proof. We will first prove that if X is a nonseparating independent set of G, then $f(X) = \sum_{v \in X}(d_G(v) - 1)$.

We proceed by induction on the size of X. Clearly, if X is a nonseparating independent set and $|X| = 0$, then $f(X) = 0 = \sum_{v \in X}(d_G(v) - 1)$.

Next suppose that for any nonseparating independent set Y of size at most $i - 1$, we have that $f(Y) = \sum_{v \in Y}(d_G(v) - 1)$ and let X be a nonseparating independent set of size i. Choose some vertex $x \in X$. The set $X - \{x\}$ is a nonseparating set of size $i - 1$, so the inductive hypothesis applies and $f(X - \{x\}) = \sum_{v \in X - \{x\}}(d_G(v) - 1)$. Then

$$f(X) = \mu(G) - \mu(G - X) = (\mu(G) - \mu(G - (X - \{x\}))) + (\mu(G - (X - \{x\})) - \mu(G - X))$$

$$= f(X - \{x\}) + (\mu(G - (X - \{x\})) - \mu(G - X)).$$

Since X is an independent set, x is not adjacent to any other member of X and so removing them from the graph does not affect the degree of x. In other words $d_G(x) = d_{G-(X-\{x\})}(x)$. Thus from graph $G - X$ to graph $G - (X - \{x\})$, the number of vertices increases by one and the number of edges increases by $d_G(x)$. Since X is a nonseparating set, the number of connected components in G, $G - (X - \{x\})$, and $G - X$ are all the same. Thus from graph $G - X$ to graph $G - (X - \{x\})$, the number of connected components does not change. Putting these numbers together, we see that $(\mu(G - (X - \{x\})) - \mu(G - X)) = d_G(x) - 1$. Thus,

$$f(X) = \left(\sum_{v \in X - \{x\}} (d_G(v) - 1) \right) + (d_G(x) - 1) = \sum_{v \in X} (d_G(v) - 1),$$

as desired. By induction, we see that if X is a nonseparating independent set of any size, then $f(X) = \sum_{v \in X}(d_G(v) - 1)$.

Next we prove the other direction: if X is not a nonseparating independent set, then $f(X) \neq \sum_{v \in X}(d_G(v) - 1)$. We know that $f(X)$ is defined as $f(X) = \mu(G) - \mu(G - X)$. Then if we list the elements of X as $x_1, x_2, ..., x_i$, we can rewrite $f(X)$ as

$$f(X) = \sum_{j=1}^{i} \left(\mu(G - \{x_1, \ldots, x_{j-1}\}) - \mu(G - \{x_1, \ldots, x_j\}) \right).$$

Notice that $\mu(G - \{x_1, \ldots, x_{j-1}\}) - \mu(G - \{x_1, \ldots, x_j\}) = d_{G-\{x_1,\ldots,x_{j-1}\}}(x_j) - 1 - \Delta c$ where Δc is the decrease in the number of connected components between graph $G - \{x_1, \ldots, x_j\}$ and graph $G - \{x_1, \ldots, x_{j-1}\}$. We know that $d_{G-\{x_1,\ldots,x_{j-1}\}}(x) - 1 - \Delta c$ is bounded above by $d_G(x) - 1$ with equality only if $\Delta c = 0$ and $d_{G-\{x_1,\ldots,x_{j-1}\}}(x_j) = d_G(x_j)$.

If X is not an independent set, then some vertex x_{j_1} will be adjacent to some other vertex x_{j_2}. Without loss of generality, let $j_1 < j_2$. Then in graph $G - \{x_1, \ldots, x_{j_2-1}\}$, vertex x_{j_2} will have a smaller degree than in graph G (since at least the neighbor x_{j_1} is missing). Thus $d_{G-\{x_1,\ldots,x_{j-1}\}}(x_j) \neq d_G(x_j)$ and strict inequality holds at least in that case: $\mu(G - \{x_1, \ldots, x_{j-1}\}) - \mu(G - \{x_1, \ldots, x_j\}) < d_G(x_j) - 1$.

If X is not a nonseparating set, then some minimal subset $A \subseteq X$ separates the graph. Suppose we order the x_is such that $A = \{x_1, \ldots, x_j\}$. Then vertex x_j will be a cut-vertex in graph $G - \{x_1, \ldots, x_{j-1}\}$. At that step, removing the vertex will change the number of connected components (i.e. $\Delta c \neq 0$). As a result, strict inequality will again hold: $\mu(G - \{x_1, \ldots, x_{j'-1}\}) - \mu(G - \{x_1, \ldots, x_{j'}\}) < d_G(x_{j'}) - 1$.

Combining this upper bound over the summation, we can bound $f(x)$ from above by $\sum_{v \in X}(d_G(v) - 1)$. Notice that this is exactly our target value. And provided that X is not a nonseparating independent set, at least one of the inequalities used to construct this bound will be strict. In that case, our target value cannot be achieved: $f(X) < \sum_{v \in X}(d_G(v) - 1)$. This concludes the proof that if X is not a nonseparating independent set, then $f(X) \neq \sum_{v \in X}(d_G(v) - 1)$.

As desired, we have shown that X is a nonseparating independent set if and only if $f(X) \neq \sum_{v \in X}(d_G(v) - 1)$. □

With that done, we can prove Lemma 2.18, which is reproduced below:

Lemma 2.18. *T' is a matching of $\hat{P}(G) = (T, f')$ if and only if the set of vertices S labeled with tags in T' is a nonseparating independent set.*

Proof. Suppose $\hat{P}(G) = (T, f')$ and $P(G) = (V, f)$.

Let $T' \subseteq T$ be any set. Then let S be the set of vertices labeled with tags in T'. The sum over all vertices in S of the vertex's degree minus one is equal to $2|T'|$. This is because each tag contributes either one degree-3 vertex to S or two degree-2 vertices. A degree-2 vertex contributes 1 to this sum (so two degree-2 vertices contribute 2) while a degree-3 vertex contributes 2 to the sum. Therefore each tag in T' contributes 2 to the sum, and so the sum in question is equal to $2|T'|$.

By Lemma 2.22, S is a nonseparating independent set if and only if $f(T')$ is equal to this sum (or in other words $f(T') = 2|T'|$). But the condition that $f(T') = 2|T'|$ is the definition of T' being a matching of (V, f).

Thus we have, as desired, that T' is a matching of $\hat{P}(G) = (T, f')$ if and only if the set of vertices S labeled with tags in T' is a nonseparating independent set. □

2.3.3 $(\{1, 2, 3\}, \mathbb{N})$-TRVB is polynomial-time solvable

We prove that $(\{1, 2, 3\}, \mathbb{N})$-TRVB is polynomial-time solvable by reducing to the Nonseperating Independent Tag Set problem.

Theorem 2.23. *$(\{1, 2, 3\}, \mathbb{N})$-TRVB is polynomial-time solvable.*

Proof. The following is a procedure for solving $(\{1, 2, 3\}, \mathbb{N})$-TRVB in polynomial time:

Take as input a multigraph G_0 whose breakable vertices each have degree at most 3. We can modify G_0 to get an equivalent graph G_1 (with respect to the TRVB problem) by inserting two unbreakable degree-2 vertices into every edge.

Next, convert graph G_1 into tagged graph G_2 as follows. Duplicate G_1 and include both copies into G_2. For each breakable degree-3 vertex in G_1, create two tags, assigning each copy of the vertex one of the tags. For each breakable degree-2 vertex in G_1, create one tag and assign it to both copies of the vertex.

Suppose G_1 had n vertices and m edges. Then use the polynomial-time algorithm for the Nonseparating Independent Tag Set problem to determine whether there exists a set of tags of size $m+1-n$ in G_2 such that the associated vertices make a nonseparating independent set. Output the answer of this subroutine as the answer of the $(\{1,2,3\},\mathbb{N})$-TRVB problem.

Next, we will show that the above procedure is correct, or in other words that the TRVB problem of G_1 can be solved if and only if G_2 contains a set of $m+1-n$ tags whose corresponding vertices form a nonseparating independent set.

Suppose first that the TRVB problem of G_1 can be solved by breaking the vertices in some set S_1. We will show that in that case G_2 contains a set of $m+1-n$ tags whose corresponding vertices form a nonseparating independent set. Suppose the solution of the TRVB problem breaks b_3 vertices of degree 3, b_2 vertices of degree 2, and b_1 vertices of degree 1. Breaking a vertex does not affect the number of edges in a graph, but increases the number of vertices by the degree minus one. Thus the final vertex count in the graph is b_2+2b_3+n. Since the final graph is a tree, the number of vertices is one more than the number of edges. Thus $b_2+2b_3+n=m+1$, or in other words $b_2+2b_3=m+1-n$.

Furthermore, applying Lemma 2.9 tells us that S_1 is a nonseparating independent set in G_1. Then consider the copies of G_1 making up G_2 and let S_2 consist of the two copies of S_1 with all degree-1 vertices removed. Since S_1 is a nonseparating independent set in G_1 we know that S_2 is a nonseparating independent set in G_2. There are b_2 tags of degree-2 vertices in S_2 (with the tags shared across the two halves of G_2) and $2b_3$ tags of degree-3 vertices in S_2 (with one tag in each half of G_2 per degree-3 vertex in S_1). Thus the total number of tags for vertices in S_2 is $b_2+2b_3=m+1-n$. The set T of these tags is a solution to the Nonseparating Independent Tag Set instance consisting of G_2 and $m+1-n$.

Next suppose that we find a set of tags in G_2 of size $m+1-n$ whose corresponding vertex set is a nonseparating independent set. Suppose that there are t_2 tags for degree-2 vertex pairs, and t_3 tags for degree-3 vertices (with $t_2+t_3=m+1-n$). Then among the two halves of G_2 (each of which is a copy of G_1), one half will have a nonseparating independent set of t_2 degree-2 vertices and at least $\frac{t_3}{2}$ degree-3 vertices.

We can convert this into a nonseparating independent set S in G_1 with t_2 degree-2 vertices and $\frac{t_3'}{2} \geq \frac{t_3}{2}$ degree-3 vertices. By Lemma 2.9, the graph formed by breaking the vertices of S in G_1 is connected. This graph has the same number of edges as G_1 (in particular m), but has a number of vertices equal to $n+2\times\frac{t_3'}{2}+t_2=n+t_2+t_3'$ (since breaking a degree-3 vertex increases the number of vertices by two and breaking a degree-2 vertex increases the number of vertices by one). The difference between the number of vertices and edges is $(n+t_2+t_3')-m \geq (n+t_2+t_3)-m=(n+m+1-n)-m=1$. Thus we see that the resulting graph is connected and has at least one more vertex than it has edges. The only way this is possible is if the resulting graph is a tree. In other words, breaking the vertices in S solves the TRVB problem on G_1.

We have shown that the TRVB instance that we wish to solve has a solution if and only if the Nonseparating Independent Tag Set instance that we actually solve has a solution.

Therefore we have described a correct algorithm. □

By Lemma 2.3, we can also conclude that

Corollary 2.24. *(B,U)-TRVB is polynomial-time solvable if $B \subseteq \{1,2,3\}$. So are Planar (B,U)-TRVB, Graph (B,U)-TRVB, and Planar Graph (B,U)-TRVB.*

2.4 Planar Graph TRVB is polynomial-time solvable without small vertex degrees

The overall purpose of this section is to show that variants of Planar Graph TRVB which disallow all small vertex degrees are polynomial-time solvable because the answer is always "no." Consider for example the following theorem.

Theorem 2.25. *If $b > 5$ for every $b \in B$ and $u > 5$ for every $u \in U$, then Planar Graph (B,U)-TRVB has no "yes" inputs. As a result, Planar Graph (B,U)-TRVB problem is polynomial-time solvable.*

Proof. The average degree of a vertex in a planar graph must be less than 6, so there are no planar graphs with all vertices of degree at least 6. Thus, if $b > 5$ for every $b \in B$ and $u > 5$ for every $u \in U$, then every instance of Planar Graph (B,U)-TRVB is a "no" instance. □

In fact, we will strengthen this theorem below to disallow "yes" instances even when degree-5 unbreakable vertices are present by using the particular properties of the TRVB problem. Note that this time, planar graph inputs which satisfy the degree constraints are possible; however, any such graph will still yield a "no" answer to the Tree-Residue Vertex-Breaking instance.

We begin with the proof idea in Section 2.4.1, and proceed through the details in Section 2.4.2

2.4.1 Proof idea

Consider the hypothetical situation in which we have a solution to the TRVB problem in a planar graph whose unbreakable vertices each have degree at least 5 and whose breakable vertices each have degree at least 6. The general idea of the proof is to show that this situation is impossible by assigning a scoring function (described below) to the possible states of the graph as vertices are broken. The score of the initial graph can easily be seen to be zero and assuming the TRVB instance has a solution, the score of the final tree can be shown to be positive. It is also the case, however, that if we break the vertices in the correct order, no vertex increases the score when broken, implying a contradiction.

Before we introduce the precise scoring mechanism, we can describe what aspects of the graph are being scored. Consider the state of the graph after breaking some number of vertices and in this graph consider one specific vertex. This vertex has several neighbors, some of which have degree 1. We can group the edges of this vertex that lead to degree-1 neighbors into "bundles" seperated by the edges leading to higher degree neighbors. For example, in Figure 2-3, the vertex shown has two bundles of size 2 and one bundle of size 3. Each bundle is given a score according to its size, and the score of the graph is equal to the cumulative score of all present bundles.

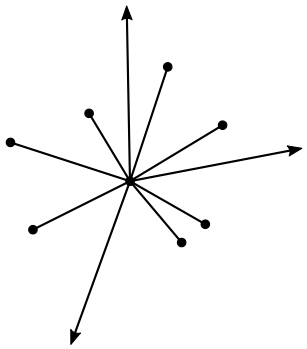


Figure 2-3: A degree-10 vertex with seven degree-1 neighbors (shown) and three other neighbors (not shown). The edges to the degree-1 neighbors form two bundles of size 2 and one bundle of size 3.

Suppose we have a set S of vertices such that breaking those vertices leaves the graph as a tree. We can choose to break these vertices in order starting on the exterior of the graph and moving inward. In particular, we can do this by repeatedly doing the following. Consider the external face of the graph. Either the graph is a tree (i.e. we have already broken all the vertices in S) or the external face is actually a cycle. Since every cycle includes a vertex in S (otherwise that cycle would remain after breaking every vertex in S), we can choose a vertex from S on the external face of the graph and break that vertex next.

Breaking the vertices of S in this order has an interesting effect on the bundles formed in the subsequent graph: in particular, since every vertex is on the external face when it is broken, every degree-1 vertex ends up in the external face when it appears. Thus all bundles are within the external face of the graph at all times.

As it turns out, the final graph (a tree) cannot have disproportionately many small bundles, so we will use the intuition that the goal of breaking the vertices is to increase bundle sizes. To apply this intuition, we consider the effect of breaking some vertex of degree d on the bundle sizes in the graph. It must be the case that any vertex in S on the external face has exactly two edges which border this face. The remaining $d-2$ edges all leave the vertex into the interior of the graph. When the vertex is broken, each of these $d-2$ edges becomes a new bundle. Thus, breaking the vertex will necessarily create $d-2$ new bundles of size 1. Applying the intuition described above, we see that creating $d-2$ bundles of size one can be thought of as the "cost" of breaking the vertex. The "benefit" of breaking the vertex is that each of the two edges which were on the external face is now each added to a bundle, thereby increasing the size of that bundle. In summary, the breaking of a vertex can increase the sizes of two bundles by 1 each at the cost of creating $d-2 \geq 4$ new bundles of size 1.

Having said that, we can now specify how we score a bundle. If a bundle has a size of 1, we assign the bundle a score of -1. Otherwise, if the bundle has size n, we assign the bundle a score of $n-1$. If we do this, then the benefit of increasing a bundle's size by 1 corresponds to a score increase of at most 2 (when increasing a bundle from size 1 to size 2). On the other hand the cost of adding a bundle of size 1 corresponds to a score decrease of 1. Thus, the effect of breaking one vertex is to increase the score by at most 2 twice and decrease the score by 1 at least 4 times. In other words, the score must either decrease or stay the same.

Then since the initial graph has no bundles (and therefore score 0), the final score of the tree that remains after breaking all vertices in S cannot be positive. On the other hand, the

final graph is a tree all of whose non-leaves have degree at least 5. It is possible to show that with the specific scoring mechanism described above, the score of such a tree is always positive. This is a contradiction, giving us our desired result.

2.4.2 Proof

In this section, we will follow the proof outline given in the previous section. We begin with a sequence of definitions leading to a formal definition of the scoring function used above.

Throughout this section, we will be considering a planar graph G whose breakable vertices each have degree at least 6 and whose unbreakable vertices each have degree at least 5.

Definition 2.26. *We say that G' is a* state *of G if we can obtain G' by breaking some vertices of G.*

We choose a particular planar embedding for G to be the canonical planar embedding for G. When a vertex is broken in some state G' of G, the new state G'' can inherit a planar embedding from G' in the natural way: all vertices and edges unaffected by the vertex-breaking are embedded in the same place while the new vertices are placed so that the order of edges around each vertex is preserved. For example, breaking the center vertex in the planar embedding shown in the left part of Figure 2-4 would yield the planar embedding shown in the center of the figure rather than the right part. Then every state of G can (via a sequence of states) inherit the canonical planar embedding for G. We will then use this embedding as the canonical embedding for the state of G. With that done, we no longer have to specify which planar embedding we are using for the states of G: we will always use the canonical planar embedding.

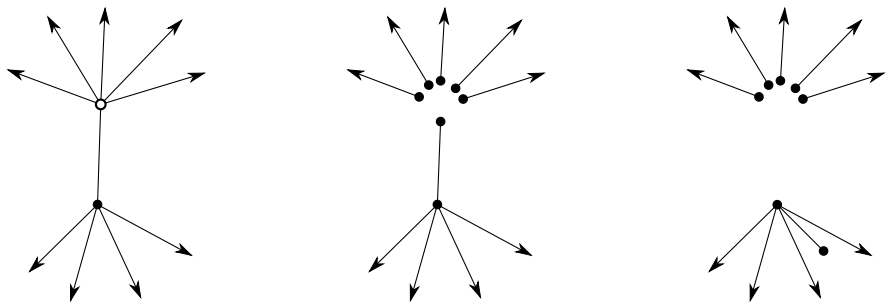


Figure 2-4: Breaking the top vertex in the first planar embedding should yield the second planar embedding rather than the third in order to maintain the order of edges around the bottom vertex.

Definition 2.27. *We call S a* contiguous set of edges at x *if S is a set of edges all sharing endpoint x and we can proceed clockwise around x starting and ending at some edge in S such that the edges encountered are exactly those in S. Then a* bundle *at vertex x is a non-empty maximal contiguous set of edges at x whose other endpoints have degree* 1*.*

Define the score of a bundle at vertex x *to be -1 if the bundle has size* 1 *and $n-1$ if the bundle has size $n > 1$. Define the* score *of x to be the cumulative score of all the bundles at x. Define the* score *of a state G' of G to be the cumulative score of all the vertices in G'.*

Suppose for the sake of contradiction that S is a set of breakable vertices in G such that breaking the vertices in S yields state T which is a tree. In other words, suppose that there exists a solution to the instance G of Planar Graph $(\{6,7,8,\ldots\},\{5,6,7,\ldots\})$-TRVB.

We begin by showing one side of the contradiction: that the score of T is positive. To do this, we introduce the following lemma about trees:

Lemma 2.28. *If T' is a tree with at least 2 vertices, then*

$$2(\textit{number of leaves of } T') + (\textit{number of degree-2 vertices in } T') > (\textit{number of edges in } T').$$

Proof. Define $n_1(T')$ to be the number of leaves in T', define $n_2(T')$ to be the number of degree-2 vertices in T', and define $n_e(T')$ to be the number of edges in T'. Then we wish to show that for any tree T' with at least 2 vertices, $2n_1(T') + n_2(T') > n_e(T')$. We will prove this by induction on the number of vertices in T'.

First consider the base case: if X' is a tree on 2 vertices, then X' contains exactly one edge (between its two vertices) so the number of leaves of X' is 2, the number of degree-2 vertices is 0 and the number of edges is 1. We see then that $2n_1(X)+n_2(X) = 2\times 2+0 = 4 > 1 = n_e(X)$ as desired.

Next suppose that for any tree X on $i-1$ vertices it is the case that $2n_1(X) + n_2(X) > n_e(X)$. Let X' be any tree on i vertices and let v be a leaf of X'. The graph $X'-\{v\}$ is a tree with $i-1$ vertices, so we can apply the inductive hypothesis: $2n_1(X'-\{v\})+n_2(X'-\{v\}) > n_e(X'-\{v\})$.

Let u be the sole neighbor of v in X. The value $2n_1 + n_2$ (twice the number of leaves plus the number of degee-2 vertices) changes as we go from $X' - \{v\}$ to X' due to the change in degree of u and the addition of v. Adding a neighbor to u either converts u from being a leaf to a degree-2 vertex, converts u from being a degree-2 vertex to a degree-3 vertex, or converts u from being a vertex of degree $n > 2$ to a vertex of degree $n+1$. In all cases, the value $2n_1 + n_2$ decreases by at most one due to the change in degree of u. The addition of the new leaf v, on the other hand, increases this value by 2. Thus the overall increase of the value $2n_1 + n_2$ when going from tree $X' - \{v\}$ to tree X' is at least 1. In other words, we have that $2n_1(X') + n_2(X') \geq 2n_1(X' - \{v\}) + n_2(X' - \{v\}) + 1$. Note also that the number of edges in X' is one more than the number of edges in $X' - \{v\}$ (i.e. $n_e(X') = n_e(X' - \{v\}) + 1$).

Putting this all together, we see that

$$2n_1(X') + n_2(X') \geq 2n_1(X' - \{v\}) + n_2(X' - \{v\}) + 1 > n_e(X' - \{v\}) + 1 = n_e(X').$$

As desired, we have shown for any tree X' with i vertices that $2n_1(X')+n_2(X') > n_e(X')$ concluding the inductive step. By induction, we have shown that for any tree T' with at least 2 vertices, $2n_1(T') + n_2(T') > n_e(T')$. □

Lemma 2.29. *The score of T is positive.*

Proof. Every vertex in T is either a vertex originally in G or a new vertex created by the breaking of some vertex in S. Vertices in G have degree at least 5 and vertices created by the breaking of a vertex have degree 1. Thus every vertex in T that is not a leaf is a vertex originally in G. Let T' be the tree formed by removing every leaf from T. Notice that the vertices in T' are exactly the vertices in $G \setminus S$.

Since every tree has at least one vertex of degree at most 1 and G does not, we know that G is not a tree. Thus $G \neq T$ and so $S \neq \emptyset$. Consider any vertex $v \in S$; v has at least 6

neighbors, none of which can be in S (since then breaking S disconnects the graph). Thus the number of vertices in T' is at least 6.

Below, we will show that the score of T is

$$-2(\text{number of edges in } T')+4(\text{number of leaves of } T')+2(\text{number of degree-2 vertices in } T').$$

But since T' is a tree with at least 6 vertices, the previous lemma applies to show that

$$2(\text{number of leaves of } T') + (\text{number of degree-2 vertices in } T') > (\text{number of edges in } T').$$

Simply rearranging (and doubling) this inequality immediately shows that the score of T is positive. Thus, all that is left is to show that the expression given above for the score of T is correct.

The score of T is the sum over all vertices x in T of the score of x. If x is not in T', then x is a leaf of T. If x has a degree-1 neighbor in T, then the connected component of x in T would consist entirely of just those two vertices and as a result, T would have no non-leaf nodes. This cannot be the case since $|T'| \geq 6$. Thus, x has no degree-1 neighbors, and therefore there are no bundles at x. As a result, the score of x is 0. Thus, the score of T is the sum over all vertices x in T' of the score of x.

For $x \in T'$, define $d(x)$ to be the degree of x in tree T'. For any vertex $x \in T'$, we can lower-bound the score of x using casework:

- $d(x) = 0$. This would imply that x is the only vertex in T', directly contradicting the fact that $|T'| \geq 6$. Thus, we can conclude that this case is impossible.
- $d(x) = 1$. The vertex x in has exactly one edge in T leading to a non-leaf neighbor. This necessarily implies that all the other edges incident on x form a bundle. Notice that x has degree at least 5 in T since it is a vertex that was originally in G. Thus, the one bundle at x has size at least 4. The score of this bundle is then at least 3, and so the score of x is also at least $3 = 4 - d(x)$.
- $d(x) = 2$. The vertex x has exactly two edges in T leading to non-leaf neighbors. These two edges seperate all of the other edges incident on x into at most two bundles. Notice that x has degree at least 5 in T since it is a vertex that was originally in G. Thus there are at least 3 edges in the (at most) two bundles at x. If there is one bundle, then the bundle has size at least 3 and score at least 2. If there are two bundles, then the total size of the two bundles is at least 3, implying that the minimum possible total score of the two bundles is 0 (which occurs in the case that one bundle has size 1 and the other has size 2). In all cases, the score of x is at least $0 = 2 - d(x)$.
- $d(x) > 2$. The vertex x has exactly $d(x)$ edges leading to non-leaf neighbors. These $d(x)$ edges seperate all of the other edges incident on x into at most $d(x)$ bundles. Each bundle has a score of at least -1, so x has a score of at least $-d(x)$.

If we use $\mathbf{1}_X$ to represent the indicator function (which outputs 1 if X is true and 0 otherwise), then the above results can be summarized as follows: the score of $x \in T'$ is bounded below by $(-d(x)) \times \mathbf{1}_{d(x)>2} + (4-d(x)) \times \mathbf{1}_{d(x)=1} + (2-d(x)) \times \mathbf{1}_{d(x)=2}$. Equivalently, we have that the score of $x \in T'$ is bounded below by $-d(x) + 4 \times \mathbf{1}_{d(x)=1} + 2 \times \mathbf{1}_{d(x)=2}$.

Adding this up, we see that the score of T is at least

$$\sum_{x \in T'} \left(-d(x) + 4 \times \mathbf{1}_{d(x)=1} + 2 \times \mathbf{1}_{d(x)=2}\right) = -\sum_{x \in T'} d(x) + 4 \sum_{x \in T'} \mathbf{1}_{d(x)=1} + 2 \sum_{x \in T'} \mathbf{1}_{d(x)=2}.$$

Since $\sum_{x \in T'} d(x)$ is the total degree of vertices in tree T', this value is twice the total number of edges in T'. The terms $\sum_{x \in T'} \mathbf{1}_{d(x)=1}$ and $\sum_{x \in T'} \mathbf{1}_{d(x)=2}$ are the number of leaves and number of degree-2 vertices in T'.

Thus the score of T is at least

$$-2(\text{number of edges in } T') + 4(\text{number of leaves of } T') + 2(\text{number of degree-2 vertices in } T').$$

As argued above, this implies our desired result: that the score of T is positive. □

Next, we proceed to the other side of the contradiction: showing that the score of T is non-positive.

To begin, we define an ordering $s_1, \ldots, s_{|S|}$ of the vertices in S as follows:

Definition 2.30. *Let $G_0 = G$. Then for $i = 1, \ldots, |S|$, define s_i and G_i as follows: Let s_i be any vertex of S that is on the boundary of the external face of G_{i-1} and let G_i be G_{i-1} with vertex s_i broken.*

Lemma 2.31. *Definition 2.30 is well defined.*

Proof. Notice that once a vertex is broken, it is no longer in the graph. Thus, it is impossible for the procedure given in Definition 2.30 to assign some element of S to be both s_i and s_j for $i \neq j$.

With that said, in order to conclude that Definition 2.30 is well defined, it is sufficient to show that at each step, a choice of s_i satisfying the conditions given in the definition is possible.

Fix $i \in \{1, \ldots, |S|\}$. Notice that G_{i-1} is not a tree since that would mean that breaking a proper subset $\{s_1, \ldots, s_{i-1}\}$ of S in G yields a tree (in which case breaking the rest of S would disconnect G). From this, we can conclude that G_{i-1} has both an external face and at least one internal face.

The points not in the external face form some set of connected regions; furthermore, this set is not empty since there is at least one internal face. Let R be any such connected region. The boundary of R must consist of a cycle of edges seperating the external face from the internal faces inside R. This cycle of edges must contain at least one vertex of S since otherwise the cycle would remain in T after breaking every vertex of S in G. Furthermore, this cycle is a subset of the boundary of the external face.

Thus, when Definition 2.30 says to choose s_i to be any vertex of S that is on the boundary of the external face of G_{i-1}, this is well defined. □

Definition 2.32. *We say that edge e is an* external edge *in G_i if e is an edge of G_i with the external face on both sides. We say that e is a* boundary edge *of G_i if the external face is on exactly one side of e. Finally, we say that e is an* internal edge *of G_i if the external face is on neither side of e.*

Lemma 2.33. *Consider any edge e incident on s_i in graph G_{i-1} and let x be the other endpoint. When converting G_{i-1} into G_i by breaking s_i,*

- *if e is a boundary edge, then e either joins one previously existing bundle at x or becomes a new bundle at x of size 1.*
- *if e is an internal edge, then e becomes a new bundle at x of size 1.*

Proof. We begin with a proof by induction that every degree-1 vertex in G_i is inside the external face for $i \in \{0, \ldots, |S|\}$. Certainly, it is the case that every degree-1 vertex in $G = G_0$ is inside the external face since G has no degree-1 vertices. Then suppose all the degree-1 vertices of G_{i-1} are inside the external face of G_{i-1} for some i. We obtain G_i from G_{i-1} by breaking vertex s_i on the boundary of the external face of G_{i-1}. As a result, the external face in G_i is equal to the union of the faces touching s_i in G_{i-1} (including the external face). Thus, every degree-1 vertex that was inside the external face of G_{i-1} is still inside the external face. Furthermore, every new degree-1 vertex (formed by breaking s_i) is also created inside the external face of G_i. By induction, we have our desired result: for $i \in \{0, \ldots, |S|\}$, every degree-1 vertex in G_i is inside the external face.

Notice that x must have at least one neighbor other than s_i that has degree more than 1 in G_{i-1} because otherwise breaking s_i would disconnect x and its neighbors from the rest of the graph. This means that if we start at edge e and go clockwise around x, we will eventually encounter some edge leading to a neighbor with degree more than 1. Call this edge e_+. Similarly, starting at edge e and going counterclockwise around x, we can let the first encountered edge leading to a neighbor with degree more than 1 be e_-. There may or may not be a bundle at x between e and e_+. Similarly there may or may not be a bundle at x between e and e_-. In any case, when s_i is broken, these bundles, if they exist, are merged, and edge e is added to the one resulting bundle.

Suppose that there is a bundle between e and e_+. This means that there are degree-1 vertices in the face that is clockwise from e_0 around x. Then since degree-1 vertices are always found inside the external face, we can conclude that the face on that side of e is the external face of G_{i-1}. Similarly, we can use the same logic to show that if there is a bundle between e and e_-, then the region on the other side of e is the external face of G_{i-1}. These results are shown in Figure 2-5.

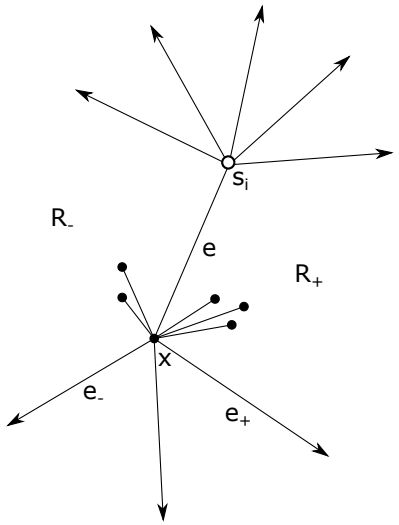


Figure 2-5: This figure shows vertices s_i and x with edge e between them and edges e_- and e_+ at x as defined in the proof. Regions R_+ and R_- are the regions on the two sides of e. The bundle shown in region R_+ at x can be present only if R_+ is the external face. Similarly, the bundle shown in region R_- at x can be present only if R_- is the external face.

Next, consider the two cases in which edge e is a boundary or internal edge. If e is an internal edge, then it has the external face on zero sides, and so neither the bundle at x

between e and e_+ nor the bundle at x between e and e_- can exist. Then when s_i is broken, edge e forms a new bundle of size 1 at x. If e is a boundary edge, then it has the external face on exactly one side. In this case, at most one of the two bundles at x (between e and e_+ or e and e_-) can exist. Then when s_i is broken, edge e either forms a new bundle of size 1 (if neither bundle existed) or is added to a previously existing bundle at x (if one of the two bundles existed). This is exactly what we wished to show. □

Lemma 2.34. *When going from G_{i-1} to G_i, the score cannot increase.*

Proof. The only vertices in G_i that are not in G_{i-1} are the degree-1 vertices which replace s_i when it is broken. The only vertex in G_{i-1} that is not in G_i is s_i. We claim that the neighbors of all of these vertices have degree not equal to 1, and therefore that each of these vertices has no bundles and thus a score of 0. Suppose for the sake of contradiction that some one of these vertices which are present in one graph but not the other has a neighbor of degree 1. This means that either a degree-1 vertex in G_i or s_i in G_{i-1} has a degree-1 neighbor. If s_i has a degree-1 neighor, then when it is broken, one of the degree-1 vertices replacing it inherits that neighbor. Thus in all cases, some two degree-1 vertices in G_i are neighbors. These two vertices form a connected component, implying that G_i is not connected. This contradicts the fact that S is a solution to the TRVB instance, so as desired, none of the vertices in question have degree-1 neighbors.

Thus, since every vertex in one of G_{i-1} and G_i but not both has a score of 0, the difference in score between G_{i-1} and G_i is equal to the cumulative difference in score over all vertices that are in both of these graphs. Suppose x is a vertex of both graphs that does not neighbor s_i. Breaking s_i does not affect x or the degrees of the neighbors of x. Thus the bundles at x remain the same in G_{i-1} and G_i. In other words, there is no change in score at vertex x.

Then to compute the difference in scores between G_{i-1} and G_i we must simply compute the cumulative difference in scores between G_{i-1} and G_i of vertices x that neighbor s_i. If the edge between s_i and x is an internal edge, then by the previous lemma, the change in bundles at x between G_{i-1} and G_i is that a new bundle of size 1 is added. This decreases the score of x by 1. If the edge between s_i and x is a boundary edge, then by the previous lemma, the change in bundles at x between G_{i-1} and G_i is either that a new bundle of size 1 is added or that the size of some one bundle is increased by 1. Thus, the score of x either decreases by 1 (if a new bundle is added), increases by 2 (if a bundle of size 1 becomes a bundle of size 2), or increases by 1 (if a bundle of size at least 2 increases in size by 1). Below, we will show that exactly two of the edges incident on s_i in G_{i-1} are boundary edges and that the rest are internal edges. Since the degree of s_i is at least 6, this means that exactly two neighbors of x will have their score increase by at most 2 and that all the other neighbors (of which there are at least 4) will have their score decrease by 1. The change in score between G_{i-1} and G_i is then an increase of at most 4 plus a decrease of at least 4. In other words, when going from G_{i-1} to G_i, the score cannot increase, which is exactly the statement of this lemma.

All that's left is to show that exactly two of the edges incident on s_i in G_{i-1} are boundary edges and that the rest are internal edges.

Consider the faces which touch at s_i in G_{i-1}. Since s_i was chosen to be on the external face, one of these faces is the external face. Let this face be F_0, and let the other faces clockwise around s_i be $F_1, F_2, \ldots F_{d-1}$ where d is the degree of s_i in G_{i-1}. Let e_j be the edge seperating face F_j from the previous face clockwise around s_i. See Figure 2-6 for an example. Finally, let x_j be the endpoint of e_j other than s_i.

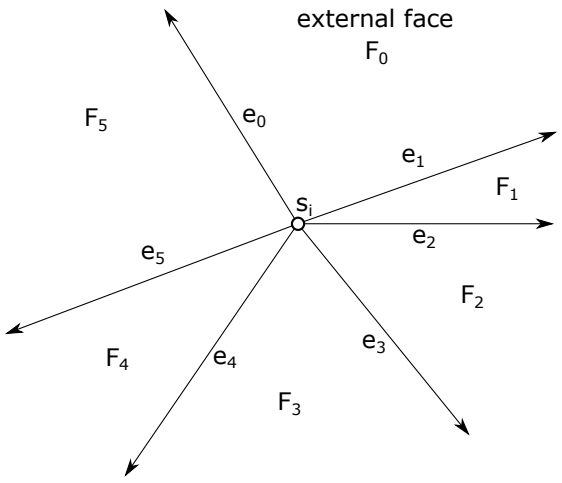


Figure 2-6: This figure shows an example vertex s_i of G_{i-1} together with the edges and faces meeting at s_i.

Since G_i is connected, there exists a path from x_0 to x_1 in G_i. This path will also exist in G_{i-1} since a path cannot use any vertices of degree 1 and every vertex in G_i whose degree is not 1 is also in G_{i-1}. There is another path in G_{i-1} from x_0 to x_1: namely the path x_0, s_i, x_1. Together, these two paths form a cycle in G_{i-1} including the two edges e_0 and e_1. Either the faces that are between e_0 and e_1 clockwise around s_i (in particular face F_0) or counterclockwise around s_i (all the other F_js) must be on the interior of this cycle. Since F_0 is the external face, we know that it cannot be on the interior of a cycle. Thus each other F_j must be on the interior of the cycle and therefore cannot equal the external face. Notice that edges e_0 and e_1 have the external face on exactly one side and that each other edge e_j has the external face on neither side. As desired, we have shown that exactly two of the edges incident on s_i in G_{i-1} are boundary edges and that the rest are internal edges. □

Lemma 2.35. *The score of T is not positive.*

Proof. $G_0 = G$ has no degree-1 vertices and therefore has no bundles. Thus, the score of G_0 is 0. By the previous lemma, the score of G_i is non-increasing as a function of i. We can immediately conclude that the score of $G_{|S|}$, the graph formed by breaking vertices $s_1, \ldots, s_{|S|}$ is not positive. But $s_1, \ldots, s_{|S|}$ are all the vertices in S, so $G_{|S|} = T$ and as desired, the score of T is not positive. □

Notice that we have seen two directly contradictory lemmas: we have shown that the score of T is both positive and not positive. By contradiction, we can conclude that S, the solution to Planar Graph $(\{6, 7, 8, \ldots\}, \{5, 6, 7, \ldots\})$-TRVB instance G, cannot exist. Thus, we have that

Lemma 2.36. *Planar Graph $(\{6, 7, 8, \ldots\}, \{5, 6, 7, \ldots\})$-TRVB is polynomial-time solvable*

Proof. We have shown that for any instance of Planar Graph $(\{6, 7, 8, \ldots\}, \{5, 6, 7, \ldots\})$-TRVB, the correct answer is "no." Thus rejecting all inputs (a polynomial-time algorithm) solves Planar Graph $(\{6, 7, 8, \ldots\}, \{5, 6, 7, \ldots\})$-TRVB. □

From this, we obtain our desired result.

Theorem 2.37. *If $b > 5$ for every $b \in B$ and $u > 4$ for every $u \in U$, then Planar Graph (B,U)-TRVB can be solved in polynomial time.*

Proof. This follows immediately from the previous lemma together with Lemma 2.3. □

2.5 Planar $(\{k\},\{4\})$-TRVB is NP-hard for any $k \geq 4$

The overall goal of this section is to prove NP-hardness for several variants of TRVB. In particular, we will introduce an NP-hard variant of the Hamiltonicity problem in Section 2.5.1 and then reduce from this problem to Planar $(\{k\},\{4\})$-TRVB for any $k \geq 4$ in Section 2.5.2. This is the only reduction from an external problem in this chapter. All further hardness results will be derived from this one via reductions between different TRVB variants.

2.5.1 Planar Hamiltonicity in Directed Graphs with all in- and out-degrees 2 is NP-hard

The following problem was shown NP-complete in [19]:

Problem 2.38. *The* Planar Max-Degree-3 Hamiltonicity Problem *asks for a given planar directed graph whose vertices each have total degree at most* 3 *whether the graph is Hamiltonian (has a Hamiltonian cycle).*

For the sake of simplicity we will assume that every vertex in an input instances of the Planar Max-Degree-3 Hamiltonicity problem has both in- and out-degree at least 1 (and therefore at most 2). This is because the existence of a vertex with in- or out-degree 0 in a graph immediately implies that there is no Hamiltonian cycle in that graph.

As it turns out, this problem is not quite what we need for our reduction, so below we introduce several new definitions and define a new variant of the Hamiltonicity problem:

Definition 2.39. *Call a vertex $v \in G$* alternating *for a given planar embedding of a planar directed graph G if, when going around the vertex, the edges switch from inward to outward oriented more than once. Otherwise, call the vertex* non-alternating*. A non-alternating vertex has all its inward oriented edges in one contiguous section and all its outward oriented edges in another; an alternating vertex on the other hand alternates between inward and outward sections more times.*

We call a planar embedding of planar directed graph G a planar non-alternating embedding *if every vertex is non-alternating under that embedding. If G has a planar non-alternating embedding we say that G is a* planar non-alternating graph*.*

Problem 2.40. *The* Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem *asks, for a given planar non-alternating directed graph whose vertices each have in- and out-degree exactly* 2*, whether the graph is Hamiltonian*

The goal of this section is to prove that this problem is NP-hard. To this purpose, consider the following definition and lemmas:

Definition 2.41. *Define* simplifying G over edge (u,v) *to be the following operation: remove all edges (u',v) and (u,v') from G and then contract edge (u,v). The resulting graph has one new vertex instead of u and v; this vertex inherits the inward oriented edges of u and inherits the outward oriented edges of v. The inward oriented edges of v and outward oriented edges of u are removed from the graph.*

Lemma 2.42. *If (u, v) is an edge of directed graph G and either u has outdegree 1 or v has indegree 1, then simplifying G over (u, v) maintains the Hamiltonicity of G.*

Proof. Let G' be the graph that results from simplifying G over edge (u, v) and let w be the vertex in G' that replaces u and v. Any Hamiltonian cycle $x_1, x_2, \ldots, x_{n-2}, u, v$ in G using edge (u, v) corresponds with Hamiltonian cycle $x_1, x_2, \ldots, x_{n-2}, w$ in G'. And any Hamiltonian cycle $x_1, x_2, \ldots, x_{n-2}, w$ in G' corresponds with Hamiltonian cycle $x_1, x_2, \ldots, x_{n-2}, u, v$ in G using edge (u, v). Thus there is a bijection between Hamiltonian cycles of G' and Hamiltonian cycles of G using edge (u, v).

But if either u has outdegree 1 or v has indegree 1, then every Hamiltonian cycle in G must use edge (u, v), and so the Hamiltonian cycles of G using edge (u, v) are all the Hamiltonian cycles of G. Thus there is a bijection between Hamiltonian cycles of G' and Hamiltonian cycles of G, and so the numbers of Hamiltonian cycles in G and G' are the same. As desired, G' is Hamiltonian if and only if G is. □

Lemma 2.43. *If (u, v) is an edge of planar non-alternating directed graph G, then simplifying G over (u, v) maintains the planar non-alternating property of G.*

Proof. Let G' be the graph that results from simplifying G over edge (u, v). Starting with a planar non-alternating embedding of G, the corresponding planar embedding of G' will also be non-alternating. We prove this below.

If x is a vertex of G that is not u or v, then in the planar non-alternating embedding x will have all the inward oriented edges in one contiguous section. The simplification of G over (u, v) will at most affect x by removing some edges incident on x. In no case does this introduce alternation of inward and outward oriented sections to x. Thus x is non-alternating in the planar embedding of G'.

If x is the new vertex introduced due to the simplification of G over (u, v), then x is non-alternating in the planar embedding of G' because (1) the inward oriented edges are all inherited from u, (2) the outward oriented edges are all inherited from v, and (3) the edges inherited from the two vertices by x can be separated into two contiguous sections.

As desired, this shows that G' is planar non-alternating. □

We apply these lemmas to prove that the Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem is NP-hard:

Theorem 2.44. *The Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem is NP-hard.*

Proof. We prove this via the following reduction from the Planar Max-Degree-3 Hamiltonicity Problem. On input a planar graph G with all in- and out-degrees 1 or 2, repeatedly identify edges (u, v) such that either u has outdegree 1 or v has indegree 1 and simplify G over (u, v). Only stop once no such edges (u, v) can be found, at which point output the resulting graph G'.

First note that this algorithm runs in polynomial time since (1) simplification is a polynomial-time operation and (2) the number of simplifications of G is bounded above by the number of vertices in G since each simplification decreases the number of vertices by 1.

Suppose the input instance G is a "no" instance of the Planar Max-Degree-3 Hamiltonicity Problem. This means that G is not Hamiltonian. By repeated application of Lemma 2.42, G' is Hamiltonian if and only if G is Hamiltonian. Thus G' is not Hamiltonian and so G' is a "no" instance of the Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem.

On the other hand, suppose the input instance G is a "yes" instance of the Planar Max-Degree-3 Hamiltonicity Problem. By repeated application of Lemma 2.42, G' is Hamiltonian if and only if G is Hamiltonian, so G' must have a Hamiltonian cycle. Below we show that all in- and out-degrees in G' are 2 and that G' is a planar non-alternating graph. Together, this is enough to imply that G' is a "yes" instance of the Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem.

Since G' has a Hamiltonian cycle, no vertex in G' can have in- or out-degree 0. Furthermore, no vertex in G' can have in- or out-degree 1 because the reduction does not stop simplifying the graph until there are no in- or out-degree 1 vertices left. Thus every in- or out-degree in G' is at least 2. When simplifying a graph over an edge, every in- or out-degree in the resulting graph is less than or equal to some in- or out-degree in the initial graph. By repeatedly applying this rule, we see that every in- and out-degree in G' is at most the largest in- or out-degree in G. But as G is a Planar Max-Degree-3 Hamiltonicity instance, the largest in- or out-degree in G is at most 2. Thus, we can conclude that every in- and out-degree in G' must be exactly 2.

By repeated application of Lemma 2.43, we know that provided the original graph G is a planar non-alternating graph, the final graph G' will be as well. But if G is a planar max-degree-3 graph, then every vertex in G is non-alternating in any planar embedding (since alternating vertices always have total degree at least 4). Thus, any planar embedding of G is a planar non-alternating embedding. We can therefore conclude that both G and G' are planar non-alternating graphs.

As desired, G is a "yes" instance of the Planar Max-Degree-3 Hamiltonicity Problem if and only if G' is a "yes" instance of the Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem. Together with the fact that the reduction runs in polynomial time, we have our desired result: the Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem is NP-hard. □

2.5.2 Reduction to Planar $(\{k\}, \{4\})$-TRVB for any $k \geq 4$

Consider the following definition:

Definition 2.45. *For $k \geq 4$, algorithm R_k defined below takes as input a planar non-alternating graph whose vertex in- and out-degrees all equal 2, and outputs an instance of Planar $(\{k\}, \{4\})$-TRVB.*

Suppose that G is the graph being given as input to R_k. Then to begin, we construct a labeled undirected multigraph M as follows.

First we build all the vertices (and vertex labels) of M. For each vertex in G, we include an unbreakable vertex in M and for each edge in G we include a breakable vertex in M. If v is a vertex or e is an edge of G, we define $m(v)$ and $m(e)$ to be the corresponding vertices in M.

Next we add all the edges of M. Fix vertex v in G. Let (u_1, v) and (u_2, v) be the edges into v and let (v, w_1) and (v, w_2) be the edges out of v. Then add the following edges to M:

- Add an edge from $m(v)$ to each of $m((u_1, v))$, $m((u_2, v))$, $m((v, w_1))$, and $m((v, w_2))$.
- Add an edge from $m((v, w_1))$ to $m((v, w_2))$.
- Add $k - 3$ edges from $m((u_1, v))$ to $m((u_2, v))$.

Finally, pick any specific vertex $\hat{v}$ in G. Let $(u_1, \hat{v})$ and $(u_2, \hat{v})$ be the edges into $\hat{v}$ and let $(\hat{v}, w_1)$ and $(\hat{v}, w_2)$ be the edges out of $\hat{v}$. We modify M by removing vertex $m(\hat{v})$ (and all incident edges), and adding the two edges $(m((u_1, \hat{v})), m((u_2, \hat{v})))$, and $(m((\hat{v}, w_1)), m((\hat{v}, w_2)))$. Call the resulting multigraph M' and return it as output of algorithm R_k.

In order to analyze the behavior of algorithm R_k, it will be helpful to have the following definition:

Definition 2.46. *We say that two edges in a planar non-alternating indegree-*2 *outdegree-*2 *graph are* conflicting *if they start or end at the same vertex. A Hamiltonian cycle in such a graph must contain exactly one out of every pair of conflicting edges.*

Lemma 2.47. *The output M' of R_k is a planar labeled multigraph whose vertices are all breakable with degree k or unbreakable with degree* 4*.*

Proof. Define all variables used in the description of R_k as defined there. Because G is planar non-alternating, we can immediately conclude that multigraph M is planar as well (see Figure 2-7 for an example).

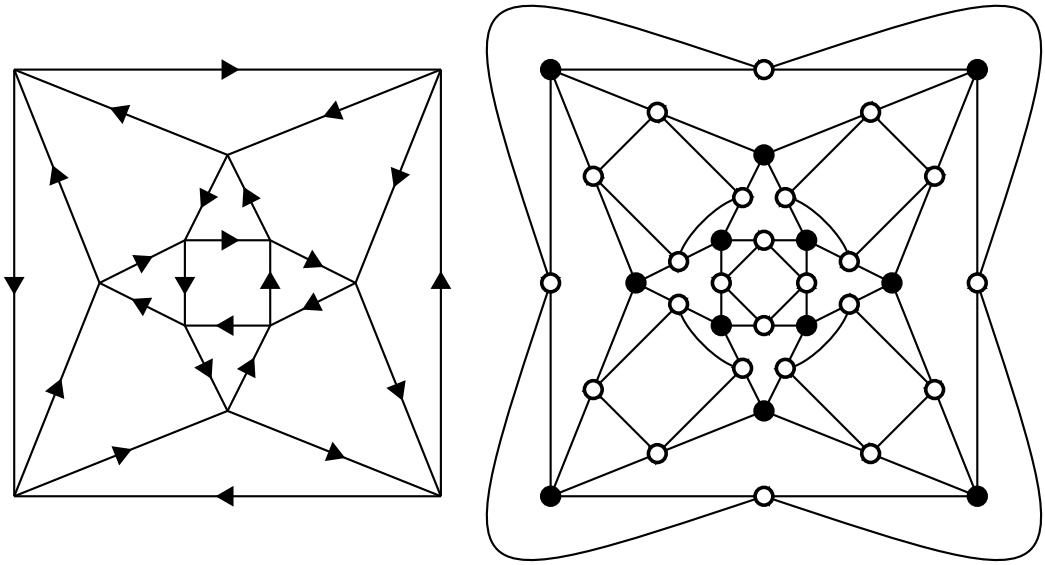


Figure 2-7: If the planar non-alternating directed graph on the left is G, then if $k = 4$ the M produced is on the right. As you can see, the planarity is maintained. If $k > 4$, then the output M remains the same except some edges are duplicated; in that case too, M is planar.

Consider any vertex $m(v)$ in M (where v is a vertex of G). This vertex has exactly four neighbors: the vertices $m(e)$ for every edge e in G that is incident on v. Furthermore, this vertex is unbreakable.

Consider any vertex $m((u, v))$ in M (where (u, v) is an edge of G). This vertex has one edge to $m(u)$, one edge to $m(v)$, one edge to $m((u, v'))$, and $k - 3$ edges to $m((u', v))$ (where (u, v') and (u', v) are the two edges in G conflicting with (u, v)). Thus the degree of this vertex is k.

This shows that M consists of only degree-4 unbreakable vertices and degree-k breakable vertices. Thus, we have shown that M has exactly those properties that we are trying to show for M': M is a planar labeled multigraph whose vertices are all breakable with degree

k or unbreakable with degree 4. All that's left is to show that the operation converting M to M' leaves these properties unchanged.

To convert M to M', vertex $m(\hat{v})$ is removed, and two edges $(m((u_1, \hat{v})), m((u_2, \hat{v})))$, and $(m((\hat{v}, w_1)), m((\hat{v}, w_2)))$ are added.

Note first that the four endpoints of these two edges are exactly the four neighbors of $m(\hat{v})$ in M. Thus, each vertex in M other than $m(\hat{v})$ has the same degree in M: either the vertex was unaffected by the change from M to M' or a single edge was removed from the vertex and a single edge was added. Therefore the vertices of M' are all breakable with degree k or unbreakable with degree 4.

Next note that the two edges added to the multigraph are both already present. Increasing the multiplicity of an edge in a multigraph does not affect the planarity of the multigraph, and neither does removal of vertices and edges. Thus, the operation transforming M into M' maintains the planarity of the multigraph.

We can conclude that we have our desired properties: M' is a planar labeled multigraph whose vertices are all breakable with degree k or unbreakable with degree 4. This can be seen for the Figure 2-7 example in Figure 2-8.

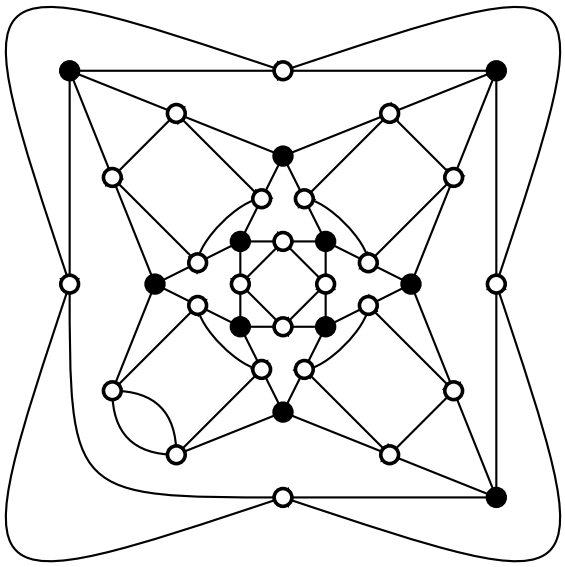


Figure 2-8: If the M produced by the algorithm is shown in Figure 2-7 (for $k = 4$), then shown here is one possible M' (where the shown multigraph is M' if $\hat{v}$ is chosen to be the bottom left vertex). Notice that M' has our desire properties: M' is a planar labeled multigraph whose vertices are all breakable with degree k or unbreakable with degree 4.

□

The following is an additional, trivial, property of R_k:

Lemma 2.48. *R_k runs in polynomial time.*

Consider the following lemma:

Lemma 2.49. *Suppose R_k outputs M' on input G and there exists a solution to the TRVB problem on M'. Then the set of edges e in G such that $m(e)$ is not broken is a disjoint cycle cover of G.*

Proof. Consider any pair of conflicting edges e_1 and e_2 in G that share endpoint v. There exists at least one edge in M between $m(e_1)$ and $m(e_2)$, and this edge is still present in M'. Thus, in order to avoid disconnecting that edge from the rest of the graph, either $m(e_1)$ or $m(e_2)$ must not be broken. M also contains a cycle on three vertices $m(e_1)$, $m(e_2)$, and $m(v)$. If $v = \hat{v}$, then the third vertex is missing in M', but in that case there is instead a cycle in M' with just $m(e_1)$ and $m(e_2)$. In any case, M' contains at least one cycle whose only breakable vertices are $m(e_1)$ and $m(e_2)$. In order for the resulting graph to be acyclic, at least one of these two vertices must be broken. This shows that in any solution to the TRVB problem on M'', exactly one out of every pair of conflicting edges (e_1, e_2) has $m(e_i)$ broken.

Consider the set C of edges e in G such that $m(e)$ is not broken. For every vertex v of G, the two edges out of v conflict and the two edges into v conflict. Since every pair of conflicting edges (e_1, e_2) has exactly one $m(e_i)$ broken, we conclude that C contains one edge that enters v and one that exists it. Thus C is a disjoint cycle cover of G, as desired. □

Based on this lemma, we can define the following correspondence:

Definition 2.50. *For any solution of TRVB instance M', define C to be the disjoint cycle cover of G consisting of edges e such that $m(e)$ is not broken in the given solution of M'.*

As per this definition, we can derive a disjoint cycle cover of G from any solution to TRVB instance M'. Similarly, for any disjoint cycle cover of G, we can derive a candidate solution (though not necessarily an actual solution) for M': simply break every vertex $m(e)$ where e is an edge of G that is not in the given disjoint cycle cover. Then for some suitable definition of "candidate solution," there is a bijection between disjoint cycle covers of G and candidate solutions of TRVB instance M'. We will show below that in fact, a disjoint cycle cover of G is actually a Hamiltonian cycle if and only if the corresponding candidate solution for M' is actually a solution. For example, see Figure 2-9.

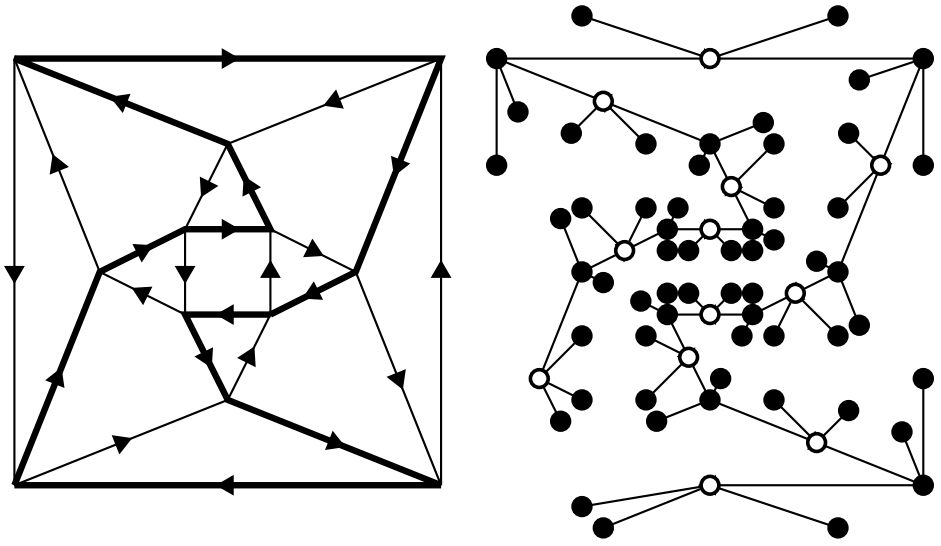


Figure 2-9: This figure shows a Hamiltonian cycle in example graph G from Figure 2-7 (left) and the corresponding solution of TRVB instance M' shown in Figure 2-8 (right).

Lemma 2.51. *Suppose R_k outputs M' on input G. If there exists a solution to the TRVB problem on M', then the corresponding cycle cover of G is actually a Hamiltonian cycle.*

Proof. Let C be the disjoint cycle cover of G consisting of edges e such that $m(e)$ is not broken in the given solution of M'. We know that C is a union of disjoint cycles and we wish to show that there is exactly one cycle in C. Let v be a vertex in G and let C_v be the cycle in C containing v. We will prove that C contains exactly one cycle by proving that C_v contains every vertex of G.

Let M'_{solved} be the solved version of M' (after breaking vertices) and let M_{solved} be a version of M in which the same vertices are broken. Consider the difference between M and M' from a connectivity standpoint. In M, vertex $m(\hat{v})$ connects its four neighbors, while in M', these neigbors are instead connected in pairs with two edges. Thus, M is at least as connected as M'. This connectivity pattern carries through to the solved versions of these multigraphs: M_{solved} is at least as connected as M'_{solved}. Since M'_{solved} is a tree, it is fully connected, and so M_{solved} is also fully connected.

From this, we see that there exists a path in M_{solved} from $m(v)$ to $m(v')$ for any vertex v' in G. This path starts in $m(v)$, ends in $m(v')$, and passes through vertices that all have degree at least 2. Therefore every vertex in this path is a vertex from the original multigraph M that was not broken. We prove below that every path in M_{solved} using only vertices originally in M which starts in $m(v)$ must end at a vertex of the form $m(x)$ where x is a vertex or edge in cycle C_v. Since there exists a path in M_{solved} using only vertices originally in M from $m(v)$ to $m(v')$, we conclude that v' is a vertex in C_v. Applying this to every vertex in G, we see that C_v is a cycle containing every vertex in G, and therefore $C = C_v$ is a Hamiltonian cycle.

Consider any path in M_{solved} using only vertices originally in M which starts in $m(v)$. We prove by induction on the path length that this path ends at a vertex of the form $m(x)$ where x is a vertex or edge in cycle C_v.

If the path length is zero, then the end vertex is $m(v)$, which is certainly of the correct form.

Next, suppose that the statement holds for all paths of length $i-1$ or less. Then given a path of length i, we can apply the inductive hypothesis to this path without the last step. Thus we have that the pre-last node in the given path is of the form $m(x)$ where x is a vertex or edge in cycle C_v. The final node in the path is a neighbor of $m(x)$ that is in M and not a broken vertex.

If x is a vertex, then the only possible non-broken neighbors of $m(x)$ are the two nodes $m(e_1)$ and $m(e_2)$ where e_1 and e_2 are the two edges into and out of x in C_v.

If x is an edge, then the neighbors of $m(x)$ are nodes of the form $m(y)$ where y is either a conflicting edge in G or an endpoint of x. But since $m(x)$ is in C_v, it was not broken, which means that the vertices in M corresponding to the conflicting edges were broken. Thus the only possible non-broken neighbors of $m(x)$ are the two nodes $m(e_1)$ and $m(e_2)$ where e_1 and e_2 are the two endpoints of x. Since x is in C_v, so are its endpoints.

We conclude that in either case, the final node in the path is of the form $m(y)$ where y is a vertex or edge in cycle C_v, proving the inductive step. By induction, any path in M_{solved} using only vertices originally in M which starts in $m(v)$ ends at a vertex of the form $m(x)$ where x is a vertex or edge in cycle C_v.

As argued above, this implies that $C = C_v$ is a Hamiltonian cycle. Thus, we have shown that if the TRVB-problem M' has a solution, then the corresponding cycle cover of G is actually a Hamiltonian cycle. □

Lemma 2.52. *Suppose R_k outputs M' on input G and there exists a Hamiltonian cycle in G. Then the corresponding candidate solution of the TRVB instance M' is a solution.*

Proof. Suppose that the Hamiltonian cycle of G is C. Then let S be the set of vertices $m(e)$ such that e is an edge of G not in C. If $m(e_1), m(e_2) \in S$, then e_1 and e_2 cannot be conflicting edges (since out of every pair of conflicting edges, one is in C). Therefore there is no edge between $m(e_1)$ and $m(e_2)$. Thus S contains no pair of adjacent vertices and is therefore an independent set in M'.

Let A be any subset of S. Define M'_A to be the multigraph derived from M' by removing the vertices in A. Every vertex in M' is either equal to $m(v)$ or adjacent to $m(v)$ for some vertex v in G. Then since $A \subseteq S$ contains only vertices of the form $m(e)$ where e is an edge of G, we conclude that every vertex in M'_A is either equal to $m(v)$ or adjacent to $m(v)$ for some vertex v in G. Then provided we show that every $m(v)$ is in the same connected component of M'_A, we will be able to conclude that M'_A is a connected graph. To show this, we prove that there exists a path in M'_A from any $m(v_1)$ to any $m(v_2)$ (with v_1 and v_2 vertices in G).

Consider the path $v_1 = x_1, x_2, \ldots, x_l = v_2$ from v_1 to v_2 in G which is part of Hamiltonian cycle C and excludes vertex $\hat{v}$. Then consider the following list of vertices:

$$m(x_1), m((x_1, x_2)), m(x_2), m((x_2, x_3)), m(x_3), \ldots, m((x_{l-1}, x_l)), m(x_l).$$

This list of vertices is a path in M, so since $m(\hat{v})$ is not in the list, it is also a path in M'. Thus we have a path in M' from $m(v_1)$ to $m(v_2)$. No vertex in this path of the form $m(x)$ where x is a vertex of G is in A simply because $A \subseteq S \subseteq \{m(e) \mid e \text{ is an edge of } G\}$. Every vertex of the form $m(e)$ where e is an edge of G has $e \in C$, and so vertex $m(e)$ is not in S (and consequentially not in A). Thus, no vertex in the path is in A, and so every vertex in the path is present in M'_A. Therefore this path is also a path from $m(v_1)$ to $m(v_2)$ in M'_A.

As described above, this allows us to conclude that M'_A is connected. We see that for any subset $A \subseteq S$, removing A from M' leaves the multigraph connected. Thus every $A \subseteq S$ is not a separating set of M', and so S is a nonseparating set of M'.

We have shown that S is a nonseparating independent set in M', so by Lemma 2.9, the multigraph that results from breaking the vertices of S in M' is connected.

Suppose G has n vertices. Then the number of edges in G is $2n$. The number of edges of G not in a Hamiltonian cycle is n, so $|S| = n$. Then the number of vertices in M is $n + 2n = 3n$, the total number of vertices and edges in G. The number of edges in M is $\frac{n \times 4 + 2n \times k}{2} = n(k+2)$. Transitioning from M to M' requires converting one vertex and four edges into zero vertices and two edges. Thus M' has $3n - 1$ vertices and $n(k+2) - 2$ edges.

Each vertex in S has degree k, so breaking the n vertices of S in M' increases the number of vertices in the resulting multigraph by $n(k-1)$. Then the multigraph that results from breaking the vertices of S in M' has $3n - 1 + n(k-1) = n(k+2) - 1$ vertices and $n(k+2) - 2$ edges. Furthermore, we showed above that this multigraph is connected. Since this multigraph is connected and has one more vertex than it has edges, we can conclude that it is a tree.

We have shown that if G has a Hamiltonian cycle, breaking the vertices in M' of set S as defined above yields a tree. Thus we have that in the case that G has a Hamiltonian cycle, the corresponding candidate solution of the TRVB instance M' is a solution. □

Theorem 2.53. *Planar $(\{k\}, \{4\})$-TRVB is NP-hard for any $k \geq 4$.*

Proof. Consider the following reduction from Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem to Planar $(\{k\},\{4\})$-TRVB. On input a graph G, we first check whether G is a planar non-alternating graph all of whose in- and out-degrees are 2. If yes, we run R_k on input G and output the result. Otherwise, we simply output any "no" instance of Planar $(\{k\},\{4\})$-TRVB.

Since R_k runs in polynomial time, the above is clearly a polynomial-time reduction. Furthermore, R_k always outputs a planar labeled multigraph whose vertices are all breakable with degree k or unbreakable with degree 4. As a result, in order to show that the above reduction is answer-preserving, it is sufficient to show that for all planar non-alternating graphs G whose in- and out-degrees are 2, G has a Hamiltonian cycle if and only if the corresponding output M' of R_k on input G, when interpreted as a TRVB instance, has a solution. This is exactly what we showed in the previous two lemmas.

Since the Planar Non-Alternating Indegree-2 Outdegree-2 Hamiltonicity Problem is NP-hard, we conclude that for any $k \geq 4$, Planar $(\{k\},\{4\})$-TRVB is NP-hard. □

2.6 Planar TRVB and TRVB are NP-complete with high-degree breakable vertices

The goal of this section is to show that Planar (B,U)-TRVB and (B,U)-TRVB are NP-complete if B contains any $k \geq 4$. To do this, we will show that Planar $(\{k\},\emptyset)$-TRVB is NP-hard for any $k \geq 4$.

Lemma 2.54. *For any $k \geq 4$, there exists a reduction from either Planar $(\{k\},\{3\})$-TRVB or Planar $(\{k\},\{4\})$-TRVB to Planar $(\{k\},\emptyset)$-TRVB.*

Proof. Below we will show that if $k \geq 4$, it is possible to simulate either a degree-4 unbreakable vertex or a degree-3 unbreakable vertex with a small gadget consisting of degree-k breakable vertices. As a result, for every $k \geq 4$, we can construct a reduction from either Planar $(\{k\},\{4\})$-TRVB or Planar $(\{k\},\{3\})$-TRVB to Planar $(\{k\},\emptyset)$-TRVB.

In particular, our reduction simply replaces every occurrence of an unbreakable degree-3 or degree-4 vertex with the corresponding gadget made of breakable degree-k vertices. Provided we can design gadgets of constant size (with respect to the size of G, not with respect to k) whose behavior is the same as the vertex they are simulating, this reduction will be correct and will run in polynomial time.

To design the gadgets, we have two cases.

For $k = 4$, we use the gadget shown in Figure 2-10. Suppose that the gadget shown was included in a Tree-Residue Vertex-Breaking instance. In order to break the cycle between P_0 and P_1 without disconnecting the edges between them from the rest of the graph, exactly one of those two vertices P_i must be broken. But then the neighbor Q_i of P_i cannot be broken without disconnecting the edge (P_i, Q_i) from the rest of the graph. On the other hand, the node Q_{1-i} cannot be broken either since breaking it would disconnect P_{1-i} from the rest of the graph. Thus any valid solution of the Tree-Residue Vertex-Breaking instance must break neither Q_i and exactly one P_i. The resulting graph connects the other four neighbors of the Q_is without forming any cycles. In other words the behavior of this gadget in a graph is the same as the behavior of an unbreakable degree-4 vertex.

For $k \geq 5$, we use the gadget shown in Figure 2-11. In this gadget, breakable vertex Q has $2a$ edges to other vertices $P_0, P_1, \ldots, P_{2a-1}$ in the gadget and $k-2a$ edges out of the gadget. In addition, there are $k-1$ edges between P_{2i} and P_{2i+1} for every i in $\{0, 1, \ldots, a-1\}$. Note

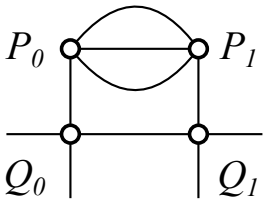


Figure 2-10: A gadget simulating an unbreakable degree-4 vertex using only breakable degree-4 vertices arranged in a planar manner.

that the degree of each vertex is k, as desired. When solving a graph containing this gadget, the cycle between P_{2i} and P_{2i+1} guarantees that exactly one of the two vertices must be broken. In order to not disconnect the other vertex from the rest of the graph, Q cannot be broken in any valid solution. Thus, provided $a > 0$, every valid solution will break exactly one P_{2i+j} with $j \in \{0, 1\}$ for each i and will not break Q. If this is done, the part of the resulting graph corresponding to this gadget will connect the $k - 2a$ external neighbors of Q to each other without forming any cycles. In other words the behavior of this gadget in a graph is the same as the behavior of an unbreakable degree-$(k - 2a)$ vertex.

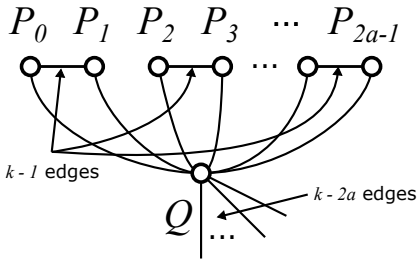


Figure 2-11: A gadget simulating an unbreakable degree-$(k-2a)$ vertex using only breakable degree-k vertices arranged in a planar manner.

Since $k \geq 5$, it is possible to choose $a > 0$ such that $k - 2a \in \{3, 4\}$. Then for every k, we are able to make a gadget to simulate either an unbreakable degree-4 vertex or an unbreakable degree-3 vertex. In all cases we can make the required gadgets, and so the reductions go through. □

We already know that Planar $(\{k\}, \{4\})$-TRVB is NP-hard from Section 2.5, so to obtain NP-hardness from the previous lemma, all that is left is to show that Planar $(\{k\}, \{3\})$-TRVB is NP-hard.

Lemma 2.55. *Planar $(\{k\}, \{3\})$-TRVB is NP-hard for any $k \geq 4$.*

Proof. We reduce from Planar $(\{k\}, \{4\})$-TRVB to Planar $(\{k\}, \{3\})$-TRVB.

Consider any unbreakable degree-4 vertex v. We can replace v with two new degree-3 unbreakable vertices u and u' with edges between the two new vertices and the neighbors of v and an edge between u and u'. If we allocate two neighbors of v to each of u and u', we succeed in making u and u' have degree 3. Note that it is possible to do this while maintaining the planarity of a multigraph. See Figure 2-12.

This pair of vertices "behaves" exactly the same as the original vertex did; in other words this change does not affect the answer to the Tree-Residue Vertex-Breaking question.

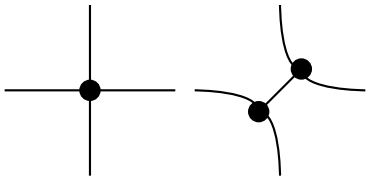


Figure 2-12: The degree-4 unbreakable vertex on the left can be simulated with two degree-3 unbreakable vertices as shown on the right while maintaining planarity.

As a result, applying this change to every unbreakable degree-4 vertex v converts a Planar $(\{k\},\{4\})$-TRVB instance into a Planar $(\{k\},\{3\})$-TRVB instance.

By Theorem 2.53, Planar $(\{k\},\{4\})$-TRVB is NP-hard for any $k \geq 4$, and so the existence of this reduction proves that Planar $(\{k\},\{3\})$-TRVB is NP-hard for any $k \geq 4$. □

Corollary 2.56. *Planar $(\{k\},\emptyset)$-TRVB is NP-hard for any $k \geq 4$.*

Proof. Lemmas 2.54 and 2.55, together with Theorem 2.53, allow us to conclude the desired result. □

Theorem 2.57. *Planar (B,U)-TRVB is NP-complete if B contains any $k \geq 4$. Also (B,U)-TRVB is NP-complete if B contains any $k \geq 4$.*

Proof. By Lemma 2.3, there is a reduction from Planar $(\{k\},\emptyset)$-TRVB to Planar (B,U)-TRVB if B contains k. Thus, if $k \geq 4$ and B contains k, then Planar (B,U)-TRVB is NP-hard. Lemma 2.3 also gives a reduction from Planar (B,U)-TRVB to (B,U)-TRVB, so we see that if $k \geq 4$ and B contains k, then (B,U)-TRVB is also NP-hard.

Using Corollary 2.5, we see that as desired, if $k \geq 4$ and B contains k, then Planar (B,U)-TRVB and (B,U)-TRVB are both NP-complete. □

2.7 Graph TRVB is NP-complete with high-degree breakable vertices

The goal of this section is to show that Graph (B,U)-TRVB is NP-complete if B contains any $k \geq 4$. To do this, we will show that Graph $(\{k\},\emptyset)$-TRVB is NP-hard for any $k \geq 4$.

Lemma 2.58. *Graph $(\{k\},\{2\})$-TRVB is NP-hard if $k \geq 4$.*

Proof. In Corollary 2.56 we saw that Planar $(\{k\},\emptyset)$-TRVB is NP-hard if $k \geq 4$. We will reduce from this problem to Graph $(\{k\},\{2\})$-TRVB.

In order to do so, we must convert a given multigraph into a graph. One way to do this is to insert two degree-2 unbreakable vertices into every edge. After doing this, there will no longer be any duplicated edges or self loops, and so the result will be graph. Furthermore, adding an unbreakable degree-2 vertex into the middle of an edge does not influence the answer to a Tree-Residue Vertex-Breaking question. Thus applying this transformation is a valid reduction.

We conclude that as desired, Graph $(\{k\},\{2\})$-TRVB is NP-hard if $k \geq 4$. □

Theorem 2.59. *Graph $(\{k\},\emptyset)$-TRVB is NP-hard if $k \geq 4$.*

Proof. In Lemma 2.58 we saw that Graph $(\{k\},\{2\})$-TRVB is NP-hard if $k \geq 4$. We wish to reduce from that problem to Graph $(\{k\},\emptyset)$-TRVB.

In order to do this, we construct a constant sized (in the size of G) gadget using degree-k breakable vertices that behaves the same as a degree-2 unbreakable vertex. Simply replacing every degree-2 unbreakable vertex with a copy of this gadget is a valid reduction, allowing us to conclude that Graph $(\{k\},\emptyset)$-TRVB is NP-hard if $k \geq 4$.

The gadget is shown in Figure 2-13. The gadget consists of $2k-2$ breakable vertices, each of degree k. Call them $P_1, P_2, \ldots, P_{k-2}$ and $Q_1, Q_2, \ldots, Q_k$. The gadget contains an edge between each pair (P_i, Q_j) and an edge between each pair (Q_i, Q_{i+1}). Finally, both Q_1 and Q_k will have one edge leaving the gadget.

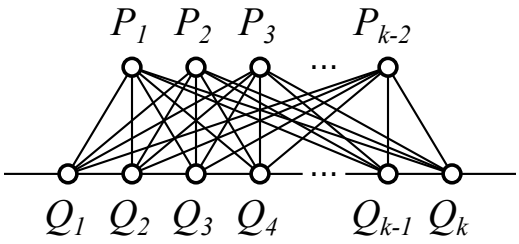


Figure 2-13: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-k vertices arranged without self loops or duplicated edges.

In a solution to this gadget, either P_1 is broken or not. If P_1 is broken, then to avoid disconnecting the edge (Q_i, P_1) from the rest of the graph, Q_i must not be broken. But (Q_1, Q_2, P_i) is a cycle for every i, so in order to avoid having that cycle in the final graph, P_i must also be broken.

If P_1 is not broken, then either Q_1 or Q_2 must be broken (due to cycle (Q_1, Q_2, P_i)). Then if Q_i is broken, P_j must not be broken in order to avoid disconnecting edge (Q_i, P_j) from the rest of the graph. This means that every P_j will not be broken. In that case, however, the existence of cycle $(Q_{i_1}, P_1, Q_{i_2}, P_2)$ guarantees that either Q_{i_1} or Q_{i_2} will be broken for every pair i_1, i_2. In other words, at most one Q_i can be unbroken. This means, however, that either both Q_1 and Q_2 or both Q_3 and Q_4 will be broken, in either case isolating an edge from the rest of the graph. Thus we see that this case is not possible.

We can conclude that the only solution to this gadget is to break all of the P_is but none of the Q_is, thereby connecting the external neighbors of Q_1 and Q_k (through the path of Q_is) without leaving any cycles or disconnecting the graph. In other words, this gadget behaves like an unbreakable degree-2 vertex, as desired.

Thus we see that the reduction goes through and Graph $(\{k\},\emptyset)$-TRVB is NP-hard if $k \geq 4$. □

Corollary 2.60. *Graph (B,U)-TRVB is NP-complete if B contains any $k \geq 4$.*

Proof. We saw in Theorem 2.59 that Graph $(\{k\},\emptyset)$-TRVB is NP-hard if $k \geq 4$, and we saw in Lemma 2.3, there is a reduction from Graph $(\{k\},\emptyset)$-TRVB to Graph (B,U)-TRVB if B contains k. Thus, if $k \geq 4$ and B contains k, then Graph (B,U)-TRVB is NP-hard. Using Corollary 2.5, we see that as desired, if $k \geq 4$ and B contains k, then Graph (B,U)-TRVB is NP-complete. □

2.8 Planar Graph TRVB is NP-hard with both low-degree vertices and high-degree breakable vertices

The goal of this section is to show that Planar Graph (B, U)-TRVB is NP-complete if (1) either $B \cap \{1, 2, 3, 4, 5\} \neq \emptyset$ or $U \cap \{1, 2, 3, 4\} \neq \emptyset$ and (2) there exists a $k \geq 4$ with $k \in B$. To do this, we will demonstrate that these conditions are sufficient to guarantee that it is possible to build small planar gadgets which behave like unbreakable degree-2 vertices. Inserting two copies of such a gadget into every edge converts a planar multigraph into a planar graph while keeping the answer to the TRVB question the same.

Below, we prove the existence of the desired gadgets.

Lemma 2.61. *There exists a planar gadget that simulates an unbreakable vertex of degree-2 built out of breakable degree-k vertices (for any $k \geq 4$) and unbreakable degree-4 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. The gadget for this theorem is shown in Figure 2-14. For each breakable vertex in this figure, there exists a cycle in the gadget containing the vertex and no other breakable vertex. To avoid leaving this cycle in the final graph, the two breakable vertices in the gadget must both be broken in a valid solution. This fully determines the solution of the gadget.

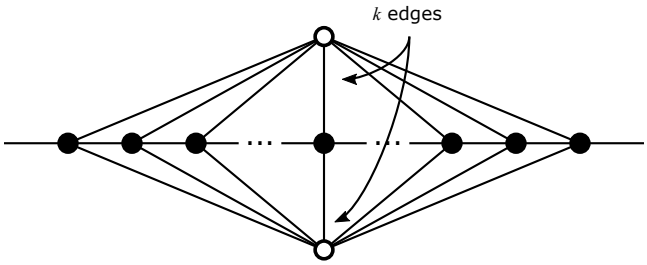


Figure 2-14: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-k and unbreakable degree-4 vertices arranged in a planar manner without self loops or duplicate edges.

Thus, if this gadget is included in a graph, the two breakable vertices must be broken, resulting in the gadget connecting the two edges that extend out to the rest of the graph. In other words, the gadget behaves like a degree-2 unbreakable vertex.

Note also that this gadget uses $k + 2$ nodes, which is constant with respect to the size of a given multigraph G. □

Lemma 2.62. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-k vertices (for any $k \geq 4$) and unbreakable degree-3 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. The gadget for this theorem is shown in Figure 2-15. The one breakable vertex in this figure is in a cycle in the gadget (with no other breakable vertex). To avoid leaving this cycle in the final graph, the breakable vertex must be broken in a valid solution. This fully determines the solution of the gadget.

Thus, if this gadget is included in a graph, the breakable vertex must be broken, resulting in the gadget connecting the two edges that extend out to the rest of the graph. In other words, the gadget behaves like a degree-2 unbreakable vertex.

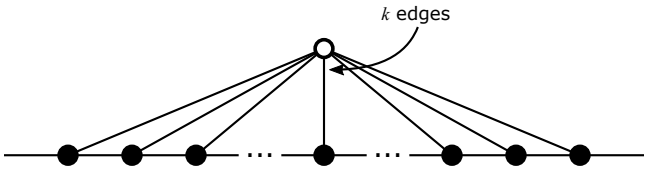


Figure 2-15: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-k and unbreakable degree-3 vertices arranged in a planar manner without self loops or duplicate edges.

Note also that this gadget uses $k + 1$ nodes, which is constant with respect to the size of a given multigraph G. □

Lemma 2.63. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-k vertices (for any $k \geq 4$) and unbreakable degree-1 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. The gadget for this theorem is shown in Figure 2-16. The one breakable vertex in this figure cannot be broken (as that would separate the unbreakable vertices from the rest of the graph).

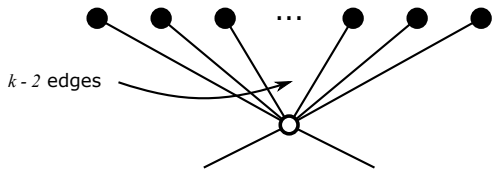


Figure 2-16: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-k and unbreakable degree-1 vertices arranged in a planar manner without self loops or duplicate edges.

Thus, if this gadget is included in a graph, the breakable vertex must not be broken, resulting in the gadget connecting the two edges that extend out to the rest of the graph. In other words, the gadget behaves like a degree-2 unbreakable vertex.

Note also that this gadget uses $k - 1$ nodes, which is constant with respect to the size of a given multigraph G. □

Lemma 2.64. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-k vertices (for any $k \geq 4$) and breakable degree-1 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. Breaking a degree-1 vertex does nothing, so breakable degree-1 vertices are essentially the same as unbreakable degree-1 vertices. Thus we can simply use the same construction as for the previous theorem. □

Lemma 2.65. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-k vertices (for any $k \geq 4$) and breakable degree-2 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. We begin by constructing the gadget shown in Figure 2-17. In this gadget, breakable vertex Q has $2a$ edges to other vertices $P_0, P_1, \ldots, P_{2a-1}$ in the gadget and $k-2a$ edges out of the gadget. In addition, there is an edge between P_{2i} and P_{2i+1} for every i in $\{0, 1, \ldots, a-1\}$. Note that the degree of Q is k and the degree of each P_i is 2. When solving a graph containing this gadget, the cycle (Q, P_{2i}, P_{2i+1}) guarantees that exactly one of the three vertices in the cycle must be broken. Q, however, cannot be broken without disconnecting P_{2i} and P_{2i+1} from the rest of the graph. Thus, provided $a > 0$, every valid solution will break exactly one P_{2i+j} out of every pair (P_{2i}, P_{2i+1}) and will not break Q. If this is done, the part of the resulting graph corresponding to this gadget will connect the $k-2a$ external neighbors of Q to each other without forming any cycles. In other words the behavior of this gadget in a graph is the same as the behavior of an unbreakable degree-$(k-2a)$ vertex.

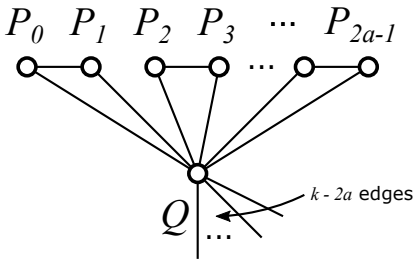


Figure 2-17: A gadget simulating an unbreakable degree-$(k-2a)$ vertex using only breakable degree-k and degree-2 vertices arranged in a planar manner without self loops or duplicate edges.

Notice that the number of nodes in the above gadget is constant with respect to the size of a given multigraph G (in particular, there are $2a+1 \leq k+1$ nodes).

Since $k \geq 4$, we can select $a > 0$ such that $k - 2a \in \{2, 3\}$. In other words, the above gadget behaves either as an unbreakable degree-2 vertex gadget or as an unbreakable degree-3 vertex gadget.

If the gadget behaves as an unbreakable degree-3 vertex gadget, then an unbreakable degree-2 vertex gadget can be built (as in the previous theorem) using breakable degree-k vertices and unbreakable degree-3 vertex gadgets. In this case, the size of the new combined gadget is at most a constant times the size of the above gadget.

Thus in all cases we can construct a gadget simulating an unbreakable degree-2 vertex using only degree-k and degree-2 breakable vertices. □

Lemma 2.66. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-3 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. The gadget for this theorem is shown in Figure 2-18. If either vertex P or vertex Q_2 is broken, then none of the others can be (since all the non-P vertices are adjacent to P and all the non-Q_2 vertices are adjacent to Q_2). If neither P nor vertex Q_2 is broken, then in order to avoid having cycles, both Q_1 and Q_3 must be broken; this however, disconnects P and Q_2 from the rest of the graph. Thus the only valid solutions of this gadget break exactly one of P and Q_2 and nothing else. In either case, the resulting graph piece connects the two edges that extend out to the rest of the graph. In other words, the gadget behaves like a degree-2 unbreakable vertex.

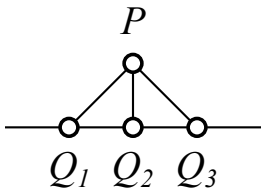


Figure 2-18: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-3 vertices arranged in a planar manner without self loops or duplicate edges.

Note also that this gadget uses 4 nodes, which is constant with respect to the size of a given multigraph G. □

Lemma 2.67. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-4 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. The gadget for this theorem is shown in Figure 2-19. Note that this is actually the same gadget as described in Theorem 2.59 for $k = 4$. Thus we have already argued the correctness of this gadget.

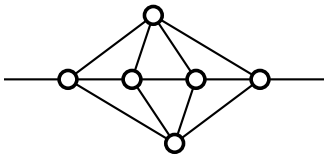


Figure 2-19: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-4 vertices arranged in a planar manner without self loops or duplicate edges.

Note also that this gadget uses 5 nodes, which is constant with respect to the size of a given multigraph G. □

Lemma 2.68. *There exists a planar gadget that simulates an unbreakable degree-2 vertex built out of breakable degree-5 vertices such that the number of nodes is constant with respect to the size of a given multigraph G.*

Proof. The gadget for this theorem is shown in Figure 2-20.

This gadget contains exactly thirty-two degree-5 vertices with two edges leaving the gadget. A choice of vertices to break within the gadget is a valid solution if either (1) the resulting graph restricted to the vertices within the gadget is a tree (with the two edges extending out of the gadget connected to this tree) or (2) the resulting graph restricted to the vertices within the gadget consists of two trees (with the two edges extending out of the gadget connected to one of these trees each). Since there are 32 degree-5 vertices with two edges out of the gadget, the number of edges in the gadget is $\frac{32\times5-2}{2} = 79$. Breaking vertices does not affect the number of edges in a graph, so the final tree or pair of trees in a valid solution of the gadget will also have 79 edges. A tree with 79 edges has 80 vertices while two trees with 79 edges have 81 vertices. Breaking one vertex increases the number of vertices by 4. Thus, it is possible to achieve the one-tree solution by breaking $\frac{80-32}{4} = 12$ vertices and it is impossible to achieve the two-tree solution.

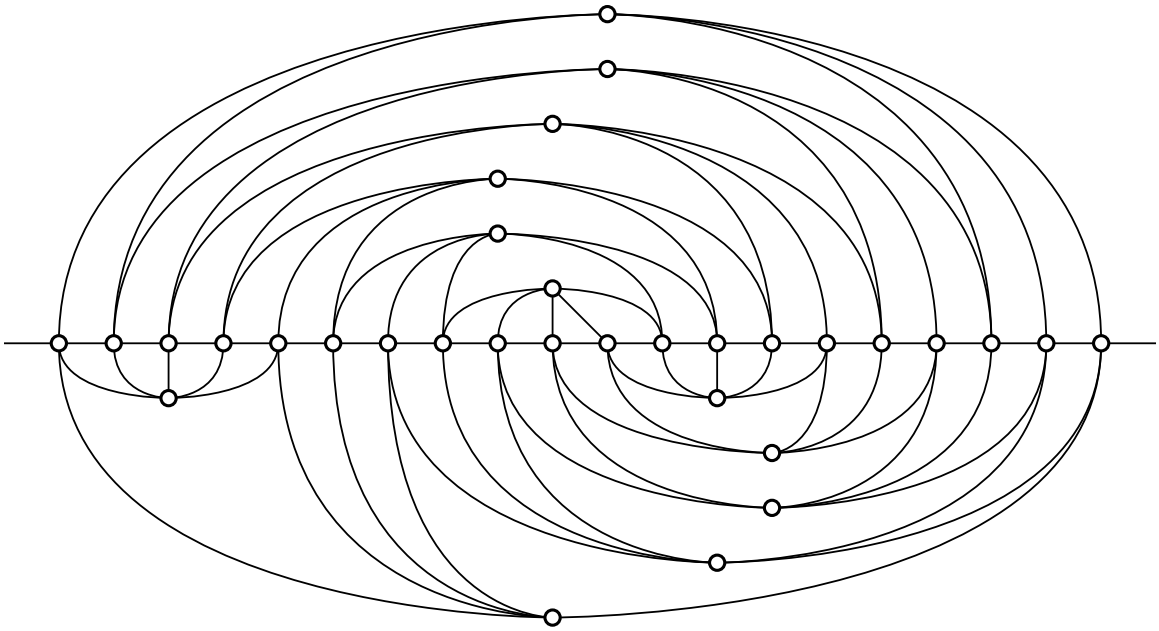


Figure 2-20: A gadget simulating an unbreakable degree-2 vertex using only breakable degree-5 vertices arranged in a planar manner without self loops or duplicate edges.

The one-tree solution corresponds with the situation in which the gadget connects the two edges that extend out to the rest of the graph. In other words, provided that the gadget can be solved, every possible solution is one under which the gadget behaves like a degree-2 unbreakable vertex. There is, however, still the question of whether the gadget can be solved. In fact, breaking every vertex above or below the center horizontal line of the gadget (and leaving the 20 vertices along the center line unbroken) is a valid solution of the gadget.

Therefore this gadget behaves like a degree-2 unbreakable vertex.

Note also that this gadget uses 32 nodes, which is constant with respect to the size of a given multigraph G. □

With these gadgets, we can now reduce from the Planar TRVB variants:

Theorem 2.69. *Planar Graph (B, U)-TRVB is NP-complete if (1) either $B \cap \{1, 2, 3, 4, 5\} \neq \emptyset$ or $U \cap \{1, 2, 3, 4\} \neq \emptyset$ and (2) there exists a $k \geq 4$ with $k \in B$.*

Proof. Suppose that for some $k \geq 4$, we have that $k \in B$ and also that either $B \cap \{1, 2, 3, 4, 5\} \neq \emptyset$ or $U \cap \{1, 2, 3, 4\} \neq \emptyset$. Then we can reduce from Planar $(\{k\}, \emptyset)$-TRVB to Planar Graph (B, U)-TRVB. Our reduction works by inserting either two degree-2 unbreakable vertices or two degree-2 unbreakable vertex gadgets (built out of vertices whose types are allowed in (B, U)-TRVB) into each edge. In either case, the resulting multigraph uses only vertices with allowed degrees (with the answer staying the same), but is now also a graph.

There are several cases:

If $B \cap \{3, 4, 5\} \neq \emptyset$, then let b be an element of $B \cap \{3, 4, 5\}$. We can build a constant size planar gadget which behaves like an unbreakable degree-2 vertex using only breakable degree-b vertices.

If $B \cap \{1,2\} \neq \emptyset$, then let b be an element of $B \cap \{1,2\}$. We can build a constant size planar gadget which behaves like an unbreakable degree-2 vertex using only breakable degree-b vertices and breakable degree-k vertices.

If $U \cap \{1,3,4\} \neq \emptyset$, then let u be an element of $U \cap \{1,3,4\}$. We can build a constant size planar gadget which behaves like an unbreakable degree-2 vertex using only unbreakable degree-u vertices and breakable degree-k vertices.

If $2 \in U$, then the problem we are reducing to allows degree-2 unbreakable vertices.

Thus, in all cases either (1) the problem we are reducing to allows degree-2 unbreakable vertices or (2) the problem we are reducing to allows vertex types with which we can build a constant sized gadget which behaves like a degree-2 unbreakable vertex.

Thus, our desired reduction from Planar $(\{k\},\emptyset)$-TRVB to Planar Graph (B,U)-TRVB is possible. Since Planar $(\{k\},\emptyset)$-TRVB is NP-hard (by Corollary 2.56), we see that Planar Graph (B,U)-TRVB is NP-hard. By Corollary 2.5, Planar Graph (B,U)-TRVB is in NP, so as desired, Planar Graph (B,U)-TRVB is NP-complete. □

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Hamiltonian Cycle in Grid Graphs

3.1 Introduction

In 2007, Arkin et al. [3] initiated a systematic study of the complexity of Hamiltonian Cycle in square, triangular, or hexagonal grid graphs, restricted to several special cases: polygonal, thin, superthin, degree-bounded, or solid grid graphs. See [3] or Section 3.2 for definitions. Table 3.1 (nonbold) summarizes the many results they obtained, including several NP-completeness results and a few polynomial-time algorithms. In this chapter, we prove that two cases left open in that paper, Hamiltonian Cycle in Square Polygonal Grid Graphs and Hamiltonian Cycle in Hexagonal Thin Grid Graphs, are NP-complete. In addition, we consider another case not considered in Arkin et al. [3], namely, *thin polygonal* grid graphs (the fusion of two special cases). We show that Hamiltonian Cycle becomes polynomially solvable in this case, for all three shapes of grid graph. Table 3.1 (bold) summarizes our new results.

In Section 3.2, we briefly define the several types of grid graphs. In Section 3.3, we show that Hamiltonian Cycle can be solved in polynomial time in the three thin polygonal grid graph cases; this is particularly challenging for hexagonal grid graphs, where the problem reduces to the polynomially solvable $(\{1,2,3\},\{1,2,3\})$-TRVB problem from Chapter 2. In Section 3.4, we prove NP-completeness of Hamiltonian Cycle in Hexagonal Thin Grid Graphs. In Section 3.5, we prove NP-completeness of the Hamiltonian Cycle in Square Polygonal Grid Graphs problem. Finally, in Section 3.6, we discuss the final remaining open

Grid	Triangular	Square	Hexagonal
General	NP-complete	NP-complete	NP-complete
Degree-bounded	$\deg \leq 3$ NP-complete	$\deg \leq 3$ NP-complete	$\deg \leq 2$ Polynomial
Thin	NP-complete	NP-complete	**NP-complete**
Superthin	NP-complete	Polynomial	Polynomial
Polygonal	Polynomial	**NP-complete**	NP-complete
Solid	Polynomial	Polynomial	Open
Thin Polygonal	**Polynomial**	**Polynomial**	**Polynomial**

Table 3.1: Complexity of Hamiltonian Cycle in grid graph variants; bold entries correspond to new results in this chapter (see [3] or Section 3.2 for definitions).

problem, Hamiltonian Cycle in Hexagonal Solid Grid Graphs.

The research in this chapter is joint work with Erik Demaine. This research was initiated during the open problem sessions of MIT's graduate class 6.890: Algorithmic Lower Bounds, Fall 2014. We thank the other participants of those sessions—in particular, Quanquan Liu and Yun William Yu—for helpful discussions and for providing a stimulating research environment.

3.2 Grid graph terminology

In this section, we introduce the definitions of several terms relating to grid graphs. We restrict our attention to only those terms and concepts relevant to the contents of this chapter. See Arkin et al. [3] for a more general overview of these concepts.

We begin with a general definition.

Definition 3.1. *The sets $\mathbb{Z}_\square$, $\mathbb{Z}_\triangle$, and $\mathbb{Z}_\hexagon$ refer to the sets of vertices of the tilings of the plane with unit-side squares, equilateral triangles, and regular hexagons. A* grid graph *is a finite graph $G = (V, E)$ where V is a subset of $\mathbb{Z}_\square$, $\mathbb{Z}_\triangle$, or $\mathbb{Z}_\hexagon$ and E is the set of pairs (u, v) of elements of V such that u and v are at a distance of 1 from each other. If $V \subset \mathbb{Z}_\square$, the grid graph is said to be a* square grid graph*. Similarly, if $V \subset \mathbb{Z}_\triangle$ then G is said to be a* triangular grid graph *and if $V \subset \mathbb{Z}_\hexagon$ then G is said to be a* hexagonal grid graph*.*

Because we are concerned with Hamiltonian Cycle, we restrict our attention to connected grid graphs with no degree-1 vertices. This does not affect the hardness of any Hamiltonian Cycle problems because all grid graphs which are disconnected or which contain a degree-1 vertex are trivially not Hamiltonian and can be easily recognized.

In order to define the grid graph properties we are interested in, we need some more terminology:

Definition 3.2. *Let G be a grid graph. Consider the faces of the graph. There is one unbounded face. The cycle bordering this unbounded face is called the* outer boundary *of G. The bounded faces of G fall into two categories. Any bounded face containing a lattice point in its interior is called a* hole*. The cycles bordering the holes of G are called the* inner boundaries *of G. The other category of bounded face is the category without lattice points in the interior; any such face must necessarily have a minimal length cycle (length 3, 4, or 6 for triangular, square, or hexagonal grid graphs) as its boundary. This type of face is called a* pixel*. Any vertex on the inner or outer boundaries is called a* boundary vertex*. All other vertices are* interior vertices*.*

The above terminology allows us to define the grid graph properties of interest:

Definition 3.3. *A* polygonal grid graph *is a grid graph $G = (V, E)$ such that every vertex in V and every edge in E belongs to a pixel and such that no vertex can be removed to merge two boundaries (See Figure 3-1, top.)*

Definition 3.4. *A grid graph is called* solid *if it has no holes, or equivalently if every bounded face is a pixel. (See Figure 3-1, middle.)*

Definition 3.5. *A grid graph is called* thin *if every vertex in the graph is a boundary vertex. Note that a thin grid graph need not be polygonal. (See Figure 3-1, bottom.)*

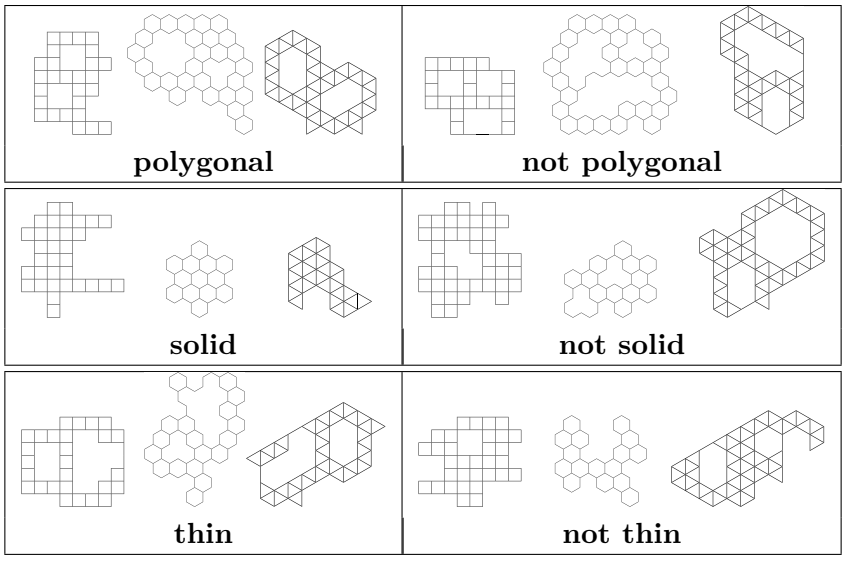


Figure 3-1: Examples of grid graphs that are or are not polygonal, solid, or thin

Now that we have defined all of the relevant terms, we can state the problems in question: the Hamiltonian Cycle in [Square/Hexagonal/Triangular] [Polygonal/Thin/Polygonal Thin] Grid Graphs problem asks whether a given [square/hexagonal/triangular] [polygonal/thin/polygonal and thin] grid graph is Hamiltonian.

3.3 Polygonal Thin Grid Graph Hamiltonian Cycle is easy

In this section, we show that the three polygonal thin grid graph Hamiltonian Cycle problems are all polynomial-time solvable. This is trivial for triangular grids and easy for square grids, but is non-trivial to show for hexagonal grids.

3.3.1 Triangular grids

Theorem 3.6 ([3])**.** *The Hamiltonian Cycle in Triangular Polygonal Thin Grid Graphs problem is polynomially solvable.*

Proof. This problem is a special case of the Hamiltonian Cycle in Triangular Polygonal Grid Graphs problem, which was shown to be polynomially solvable in [3]. □

3.3.2 Square grids

We prove below that

Theorem 3.7. *Every polygonal thin square grid graph is Hamiltonian.*

and therefore conclude that

Corollary 3.8. *There exists a polynomial time algorithm which decides the Hamiltonian Cycle in Square Polygonal Thin Grid Graphs problem (the "always accept" algorithm).*

To prove Theorem 3.7, we will provide a polynomial-time algorithm for finding a Hamiltonian cycle in a polygonal thin square grid graph, prove that if the algorithm produces an output then the output is a Hamiltonian cycle, and prove that following the algorithm is always possible.

First, the algorithm: Suppose the set of pixels in input graph G is P. Initialize S to be the empty set. Then repeat the following step until $P - S$ contains no cycles of pixels: identify a cycle of pixels in $P - S$, find a pixel p in this cycle such that exactly two pixels in G neighbor p and the two neighboring pixels are on opposite sides of p, and add p to S. Once this loop is finished, let $T = P - S$ be the set of pixels in G but not in S. Treating T as a region, output the boundary of that region as a Hamiltonian cycle.

Clearly, this algorithm is a polynomial-time algorithm. The only questions are (1) whether the output is actually a Hamiltonian cycle if the algorithm succeeds and (2) whether a given cycle of pixels always contains a pixel p such that exactly two pixels in G neighbor p and the two neighboring pixels are on opposite sides of p. We prove that the answer to both these questions is "yes" below:

Lemma 3.9. *Provided the given algorithm succeeds at each step on input G, the final output will be a Hamiltonian cycle in G.*

Proof. Since G is connected, the pixels in P are connected as well. At every step of the algorithm, $P - S$ remains connected since the only pixel added to S (and therefore removed from $P - S$) is a pixel in a cycle with no neighbors outside the cycle. Furthermore, the final value of $P - S$ (also known as T) will be acyclic since that is the terminating condition of the loop. Thus T is connected and acyclic. In other words T is a tree of pixels. As a result, the region defined by T is connected and hole-free. Therefore the boundary of T is one cycle. All that is left to show is that every vertex in G is on this boundary.

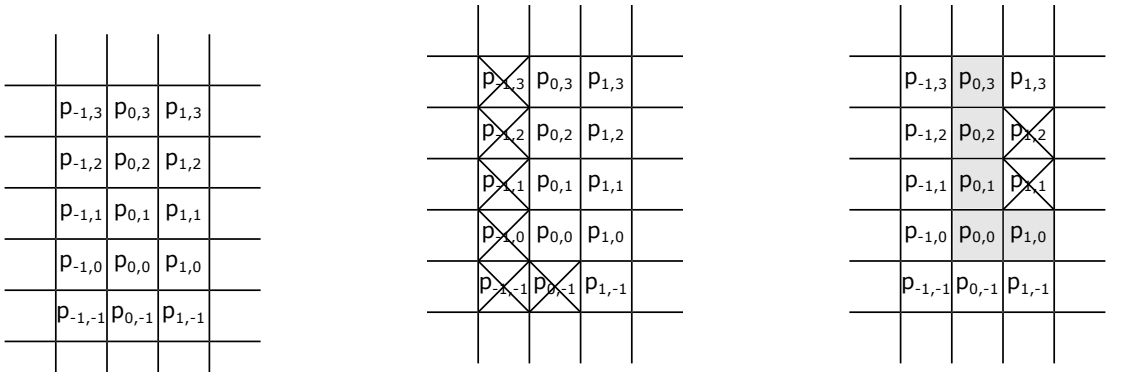
Consider any vertex v in G. G is a thin grid graph, so every vertex, including v, is on the boundary of G. Then provided v is adjacent to some pixel in T, we also have that v is on the boundary of T. Thus we need to show that every vertex is adjacent to at least one pixel in T.

Consider any pair of adjacent pixels p_1 and p_2 such that each of the two pixels has exactly two neighbors. As soon as one of these pixels is added to S (if this ever occurs), the other will forevermore have at most one neighbor in $P - S$. As a result, this second pixel will never be in a cycle of pixels in $P - S$. Then this second pixel will never itself be added to S, or in other words at most one of p_1 and p_2 will be added to S. Thus, the final value of the set S will contain no two adjacent pixels, or in other words every pixel adjacent to a pixel in S will be in T.

But if S contains pixel p then every vertex adjacent to p is also adjacent to one of the two neighbors of p (since the two neighbors must be on opposite sides of p). Since these neighbors are in T, we see that every vertex adjacent to a pixel in S is also adjacent to a pixel in T.

Since the graph is polygonal, every vertex in the graph is adjacent to some pixel: either a pixel in T or a pixel in S. In either case, we can conclude that the vertex is adjacent to a pixel in T, and therefore, as argued above, the boundary of T is a Hamiltonian cycle in G. □

Lemma 3.10. *For any cycle of pixels C in a polygonal thin grid graph G, there exists a pixel p in that cycle such that exactly two pixels in G neighbor p and the two neighboring pixels are on opposite sides of p.*



(a) The naming scheme given to the pixels in the plane.

(b) Pixels that cannot be in C by definition of $p_{0,0}$ are crossed out.

(c) Pixels that must be in C are shaded gray and those not in G are crossed out.

Figure 3-2

Proof. Consider the leftmost column of pixels which contains any pixels in C and let $p_{0,0}$ be the bottom-most pixel of C in this column. Assign x and y coordinates to the pixels in $\mathbb{Z}_\square$ so that $p_{0,0}$ has coordinates $(0, 0)$ and the coordinates increase as we go up and to the right. See Figure 3-2a.

By definition of $p_{0,0}$, we know that $p_{-1,i}$ is not a pixel in C for any i and neither is $p_{0,-1}$. These pixels are crossed out in Figure 3-2b.

But C is a cycle so $p_{0,0}$ must have exactly two neighbors in C. Therefore $p_{1,0}$ and $p_{0,1}$ must both be in C. Then in order for G to be thin, pixel $p_{1,1}$ cannot be in G (nor in C). $p_{0,1}$ is a pixel in C and therefore must have two neighbors in C. Since neither $p_{-1,1}$ nor $p_{1,1}$ are pixels in C we can conclude that these two neighbors must be $p_{0,0}$ and $p_{0,2}$. In particular, $p_{0,2}$ must be a pixel in C.

Suppose for the sake of contradiction that $p_{1,2}$ is a pixel in G. Then $p_{1,2}$, $p_{0,2}$, $p_{0,1}$, $p_{0,0}$, and $p_{1,0}$ are all pixels in G. As a result, all four vertices on the boundary of pixel $p_{1,1}$ are in G, and so since G is an induced subgraph, the edges between these vertices are in G as well. As a result, we can conclude that pixel $p_{1,1}$ is in G, which is a contradiction. Thus $p_{1,2}$ is not a pixel in G.

Pixel $p_{0,2}$ is in C and therefore must have two neighbors in C. Since neither $p_{-1,2}$ nor $p_{1,2}$ is in C we can conclude that these two neighbors must be $p_{0,1}$ and $p_{0,3}$. In particular, $p_{0,3}$ must be in C.

We have shown that $p_{0,0}$, $p_{0,1}$, $p_{0,2}$, and $p_{0,3}$ are all pixels in C and that $p_{1,1}$ and $p_{1,2}$ are not pixels in G. See Figure 3-2c. Since G is thin, either $p_{-1,1}$ or $p_{-1,2}$ must be a pixel not in G. Then for some $i \in \{1, 2\}$ we have that $p_{0,i}$ is a pixel in C, $p_{0,i\pm1}$ are pixels in C, and $p_{\pm1,i}$ are pixels not in G.

That pixel $p_{1,i}$ is the pixel p we wished to find: a pixel in C such that exactly two pixels in G neighbor p and the two neighboring pixels are on opposite sides of p. □

3.3.3 Hexagonal grids

Recall the Tree-Residue Vertex-Breaking (TRVB) problem from Chapter 2:

Problem 3.11. *The Tree-Residue Vertex-Breaking problem (TRVB) asks for a given multigraph G in which every vertex is labeled as "breakable" or "unbreakable" whether there exists a subset of the breakable vertices such that "breaking" those vertices results in a tree.*

Here the operation of breaking *a vertex in a multigraph (shown in Figure 3-3) results in a new multigraph by removing the vertex, adding a number of new vertices equal to the degree of the vertex in the original multigraph, and connecting these new vertices to the neighbors of the vertex in a one-to-one manner.*

(a) Before. (b) After.

Figure 3-3: An example of breaking a vertex

In particular, consider the following variant of TRVB:

Theorem 3.12. *The* $(\{1,2,3\},\{1,2,3\})$*-TRVB (Max-Degree-*3 *TRVB) problem asks the same question as the TRVB problem, but restricts the inputs to be graphs whose vertices each have degree at most* 3*. We showed that* $(\{1,2,3\},\{1,2,3\})$*-TRVB is polynomial-time solvable in Chapter 2.*

In this section, we will show that

Theorem 3.13. *There exists a polynomial-time reduction from the Hamiltonian Cycle in Hexagonal Polygonal Thin Grid Graphs problem to* $(\{1,2,3\},\{1,2,3\})$*-TRVB.*

and therefore that

Corollary 3.14. *The Hamiltonian Cycle in Hexagonal Polygonal Thin Grid Graphs problem is polynomial-time solvable.*

We prove Theorem 3.13 with the following reduction: On input a hexagonal polygonal thin grid graph G, construct the graph G' whose vertices are pixels and whose edges connect adjacent vertices. Label a vertex in G' as breakable if it has degree-3. Otherwise label the vertex unbreakable. Output the resulting labeled graph.

To prove the correctness of this reduction, we first consider the behavior of a Hamiltonian cycle in the local vicinity of a pixel, then narrow down the possibilities using non-local constraints, and finally use the global constraints imposed by the existence of a Hamiltonian cycle to characterize the hexagonal polygonal thin grid graphs with Hamiltonian cycles. We will show using this characterization that a hexagonal polygonal thin grid graph is Hamiltonian if and only if the corresponding reduction output is a "yes" instance of $(\{1,2,3\},\{1,2,3\})$-TRVB.

Lemma 3.15. *In a hexagonal polygonal thin grid graph, the only possible behaviors for a Hamiltonian cycle near a given pixel in the graph look like one of the diagrams in Figures 3-4b, 3-4c, 3-4d, or 3-4e depending on the number of other pixels adjacent to the pixel in question.*

Proof. Consider the pattern of pixels in Figure 3-4a. In any grid graph containing three pixels in this arrangement, the circled vertex is not on the boundary. Thus, because every vertex must be on the boundary of a thin grid graph, the three-pixel pattern does not occur in any hexagonal thin grid graph.

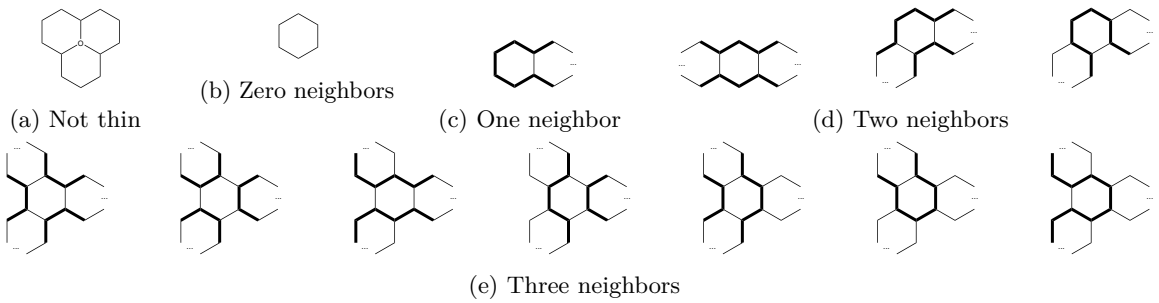


Figure 3-4: Local constraints on hexagons

Next consider a single pixel. It can have up to 6 neighboring pixels, but in order to avoid the three pixel arrangement from Figure 3-4a, it will have at most 3 neighbors.

If a pixel has zero neighbors (i.e., Figure 3-4b), it is a single connected component of the graph. Because the graphs we consider are connected, that means that the pixel is the entire graph. In that case, there is a Hamiltonian cycle (consisting of the whole graph). Since in this case we can easily solve the Hamiltonian Cycle problem, we restrict our attention to other cases.

If a pixel has exactly one neighbor, the cycle must pass through it as shown in Figure 3-4c (up to rotation).

If a pixel has exactly two neighbors, they can be arranged (up to rotation) in two ways. If the two neighboring pixels are opposite, the cycle must pass through the pixel as shown in Figure 3-4d (left). In the other arrangement, there are two possibilities as shown in Figure 3-4d (right).

Finally, there is exactly one way to arrange a pixel with three neighboring pixels, and the cycle can pass through this type of pixel in seven different ways. This is shown in Figure 3-4e.

The different possibilities of how to arrange the cycle in a given diagram were computed by exhaustive search subject to the constraints that (1) every vertex must be in the cycle and (2) exactly two edges next to each vertex must be in the cycle. □

Lemma 3.16. *In a hexagonal polygonal thin grid graph, the only possible behaviors for a Hamiltonian cycle near a given pixel look like one of the diagrams in Figure 3-5e. Furthermore, the final situation in Figure 3-5e necessarily leads to the situation shown in the rightmost part of Figure 3-5f.*

Proof. Suppose that we have three pixels arranged as shown in the leftmost part of Figure 3-5a with the three bold edges definitely included in the Hamiltonian Cycle. Because the circled vertex in the second part of the figure has to have two edges, we can conclude that the situation must be as shown in the third part of the figure. But the bottom right edge must be part of a pixel, and because every vertex must be on the boundary, only one of the two possible pixels will work. This yields the situation in the fourth part of the figure. Consider the circled vertex in the fifth part of the figure. That vertex must have two edges in the Hamiltonian cycle touching it. One is already accounted for, so we just have to decide on the other.

Consider for the sake of contradiction that the chosen edge was as shown in the first part of Figure 3-5b. Then because of the circled vertex in the second part of the figure, the

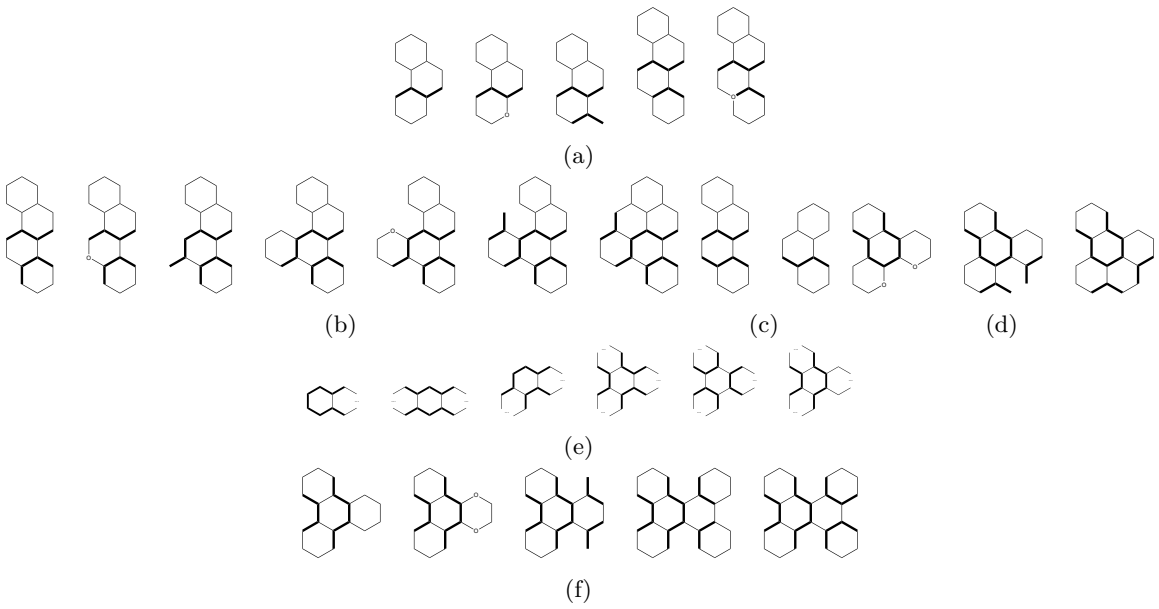


Figure 3-5: Less local constraints on hexagons

situation must be as shown in the third part of the figure. Again, every edge must be part of a pixel, and again only one of the two possible pixels would yield a thin graph. Thus we arrive at the situation in the fourth part of the figure. Because of the circled vertex in the fifth part of the figure, we arrive at the situation in the sixth. But then the dotted edge in the seventh figure must also be in the graph (because no two adjacent vertices can exist in the graph without the edge between them also being in the graph). This, however, yields a graph that is not thin. Thus we have a contradiction.

Therefore, what must actually have happened is shown in the left side of Figure 3-5c. Looking only at the bottom three pixels in this new situation and ignoring some of the bold lines, we have the situation in the right half of the figure. Note, however, that this situation is identical (up to reflection and translation down) to the situation at the start of Figure 3-5a. Thus, having the pattern in the first part of Figure 3-5a necessitates another copy of the same pattern lower down (and flipped). That in turn necessitates another copy lower down, which necessitates another copy lower down, etc... In other words, no finite graph can contain the pattern shown at the start of Figure 3-5a.

As a result of this, many of the "possible" local solutions at a pixel are actually not allowed.

We can restrict the possibilities even more, however. Consider the penultimate scenario from Figure 3-4e. Under this scenario, consider the circled vertices in the first part of Figure 3-5d. The constraints at those vertices lead to the scenario in the center part of that figure, which in turn leads to the existence of the edge added in the final part of the figure. This contradicts the fact that the grid graph must be thin, so the actual list of possible local solutions does not include the penultimate scenario from Figure 3-4e.

Then the restricted list of possibilities is as shown in Figure 3-5e where we include only those local solutions which omit the pattern at the start of Figure 3-5a and where we exclude the penultimate scenario from Figure 3-4e.

Finally, we wish to show that the last situation listed in Figure 3-5e directly leads to the situation shown in the last part of Figure 3-5f. The first part of Figure 3-5f is exactly

the final situation listed in Figure 3-5e. Consider the circled vertices in the second part of Figure 3-5f. The constraints at these vertices lead to the situation shown in the third part of Figure 3-5f. Next, due to the thin and polygonal properties, the pixels added in the penultimate part of Figure 3-5f must be present in the graph. Consider the rightmost pixel with three neighbors. We have a list of situations that are possible in the vicinity of a pixel with three neighbors (in Figure 3-5e), and only one of those possibilities matches the already chosen edges. Thus, we must have the situation shown in the last part of Figure 3-5f. □

Lemma 3.17. *There exists a Hamiltonian cycle in a hexagonal polygonal thin grid graph if and only if there exists a tree of pixels such that every pixel with fewer than three neighbors is in the tree and such that at least one pixel out of every pair of adjacent degree-3 pixels is in the tree.*

Proof. First suppose there exists a Hamiltonian cycle in a hexagonal polygonal thin grid graph.

Based on the local solutions for degree-1 and degree-2 pixels, whenever one pixel with degree at most two is adjacent to another, the two pixels are on the same side of the cycle (both inside or both outside). Based on the local solutions for degree-3 pixels (and based on the specific situation that the final situation listed in Figure 3-5e necessarily leads to), whenever two degree-1 or degree-2 pixels are adjacent to the same degree-3 pixel, the two pixels are on the same side of the cycle. Finally, whenever two degree-3 pixels are neighbors, their other four neighbors are all degree-1 or degree-2 pixels and are all on the same side of the cycle. All together, this implies that all pixels with at most two neighbors must be on the same side of the cycle. And because the unbounded face is outside the cycle, the faces next to it—pixels—must be inside the cycle.

We can conclude that all pixels with two or fewer neighbors are always inside the cycle. This immediately implies that all holes are outside the cycle.

Clearly, the pixels inside the Hamiltonian cycle are connected. Furthermore, they are acyclic because a cycle inside the Hamiltonian cycle would imply either that the Hamiltonian cycle contains a hole or that the graph contains an interior vertex. Thus the pixels inside the Hamiltonian cycle form a tree.

In addition, it is easy to verify that whenever two degree-3 pixels are adjacent, they are not both outside the Hamiltonian cycle.

Next suppose that there exists a tree of pixels such that every pixel with fewer than three neighbors is in the tree and such that at least one pixel out of every pair of adjacent degree-3 pixels is in the tree. Then consider the perimeter of this tree of pixels. We claim that the perimeter is a Hamiltonian cycle. Clearly, the perimeter is a cycle, so the only question is whether it touches every vertex. Every vertex is either between two degree-3 vertices or adjacent to at least one degree-2 or degree-1 pixel. In either case, the assumptions are sufficient to show that the vertex is on the perimeter of at least one pixel in the tree. Because the graph is thin, there are no interior vertices; in other words, every vertex on the perimeter of a pixel in the tree is also on the perimeter of the entire tree of pixels. Thus the perimeter of the tree of pixels hits every vertex and is therefore a Hamiltonian cycle.

We have now shown both directions of the desired statement. □

Lemma 3.18. *Suppose G is a hexagonal polygonal thin grid graph and G' is the output of the reduction on input G. Then G' is a "yes" instance of $(\{1,2,3\},\{1,2,3\})$-TRVB if and only if there exists a tree of pixels in G such that every pixel with fewer than three neighbors*

is in the tree and such that at least one pixel out of every pair of adjacent degree-3 pixels is in the tree.

Proof. First suppose that G' is a "yes" instance of $(\{1,2,3\},\{1,2,3\})$-TRVB. Then let B be the set of breakable vertices in G' whose breaking yields a tree. Vertices in G' correspond with pixels in G, so define T to be the set of pixels in G that do not correspond to vertices in B.

Breaking a vertex in G' corresponds to removing the vertex and then adding some number of degree-1 vertices. The result of breaking the vertices of B in G' is a tree G'_{break}. If instead of breaking the vertices we simply removed them, we would get a graph G'_{remove} which could instead be obtained by removing the degree-1 vertices that were added during the breaking operation. Since removing degree-1 vertices from a tree yields a tree, we can conclude that G'_{remove} is a tree. But G'_{remove} has as vertices the pixels of T and as edges the adjacencies between pixels. Thus T forms a tree of pixels.

By the way the reduction was defined, every breakable vertex corresponds to a pixel in G of degree-3. Thus B contains only vertices of degree-3 and so T contains every pixel that has fewer than three neighbors.

Consider any pair of adjacent degree-3 pixels. If the vertices corresponding to both pixels were in B then both vertices would be broken, leading to the edge between those two vertices being disconnected from the rest of the graph. Since breaking the vertices of B in G' yields a tree, this is not the case. Thus, at least one of the two vertices is not in B, and so at least one of the pair of pixels is in T.

We have shown that T is a tree of pixels in G such that every pixel with fewer than three neighbors is in the tree and such that at least one pixel out of every pair of adjacent degree-3 pixels is in the tree. Thus whenever G' is a "yes" instance of $(\{1,2,3\},\{1,2,3\})$-TRVB such a tree of pixels must exist.

Next we prove the other direction. Suppose that T is a tree of pixels in G such that every pixel with fewer than three neighbors is in the tree and such that at least one pixel out of every pair of adjacent degree-3 pixels is in the tree. Define B to be the set of breakable vertices in G' corresponding to pixels of G that are not in T.

Consider the graph G'_{remove}, which is constructed from G' by removing every vertex in B. The vertices of this graph are the pixels in T and the edges are adjacencies of pixels. Thus, since T is a tree of pixels, G'_{remove} is a tree.

Also consider the graph G'_{break} that results from breaking the vertices in B. Breaking a vertex can be accomplished by removing it and then adding some number of degree-1 vertices. Therefore the graph G'_{break} could also be constructed from G'_{remove} by adding some number of degree-1 vertices. If every vertex that is added has as its sole neighbor a vertex from G'_{remove} then the addition of these degree-1 vertices maintains the tree property, allowing us to conclude that G'_{break} is a tree.

Note that by the definition of T, no two vertices in B are adjacent. Suppose v' is a degree-1 vertex created to neighbor vertex $u \in G'$ during the breaking operation of vertex $v \in B$. Since B contains no adjacent pairs of vertices, $u \notin B$, and therefore u is never broken when constructing G'_{break}. Thus the sole neighbor of v' in G'_{break} is u, which is a vertex from G'_{remove}. As stated above, applying this logic to every v' allows us to assert that G'_{break} is a tree, and therefore (since the maximum degree of vertices in G' is at most 3) that G' is a "yes" instance of $(\{1,2,3\},\{1,2,3\})$-TRVB.

We have shown both directions, thus proving the desired equivalence. □

Putting the last two lemmas together, we conclude that the given reduction is correct.

3.4 Hamiltonian Cycle in Hexagonal Thin Grid Graphs is NP-complete

In this section, we show that the Hamiltonian Cycle in Hexagonal Thin Grid Graphs problem is NP-complete. Membership in NP is trivial, while NP-hardness follows via reduction from Planar $(\{6\}, \emptyset)$-TRVB.

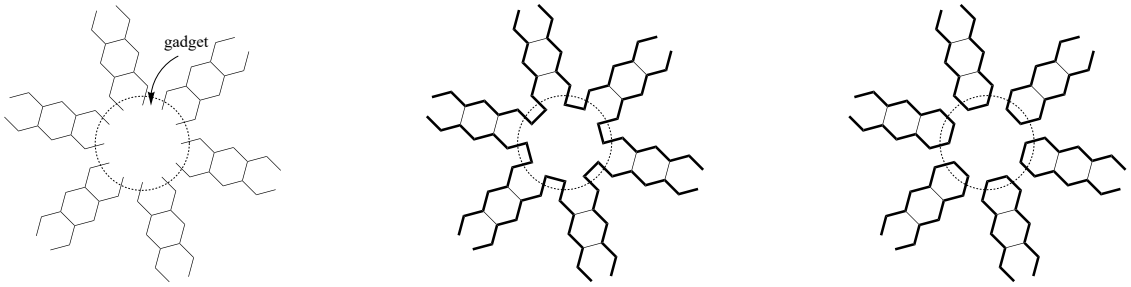
Recall the definition of the Tree-Residue Vertex-Breaking problem:

Problem 3.11. *The Tree-Residue Vertex-Breaking problem (TRVB) asks for a given multigraph G in which every vertex is labeled as "breakable" or "unbreakable" whether there exists a subset of the breakable vertices such that "breaking" those vertices results in a tree.*

Here the operation of breaking *a vertex in a multigraph (shown in Figure 3-3) results in a new multigraph by removing the vertex, adding a number of new vertices equal to the degree of the vertex in the original multigraph, and connecting these new vertices to the neighbors of the vertex in a one-to-one manner.*

We showed in Chapter 2 that Planar $(\{6\}, \emptyset)$-TRVB—a variant of TRVB in which the inputs is restricted to be a planar multigraph whose vertices are each breakable vertices of degree exactly 6—is NP-complete.

The main work of our reduction is constructing a gadget to simulate a degree-6 breakable vertex. The desired behavior of the gadget is shown in Figure 3-6. If we define a wire to be a path of pixels then the idea of the gadget is to connect 6 incoming wires in such a way that the local constraints on a hypothetical Hamiltonian cycle allow only two possible solutions within the gadget. In one of the locally allowed solutions (Figure 3-6b) the regions inside the six wires are connected through the gadget while in the other (Figure 3-6c) the regions are all disconnected at the gadget. Note that the gadget shown in Figure 3-6 is only a schematic and cannot be used as the actual gadget for the reduction because it does not lie on the hexagonal grid.



(a) A degree-6 breakable gadget connects 6 wires with only two local solutions

(b) One solution of the gadget connects the regions inside the wires

(c) The other solution disconnects the regions inside the wires

Figure 3-6: Desired behavior of a degree-6 breakable vertex gadget

Below, we will (1) provide and analyze a reduction from Planar $(\{6\}, \emptyset)$-TRVB under the assumption of the existance of a degree-6 breakable vertex gadget and (2) provide the gadget and prove that it has the desired behavior.

3.4.1 Reduction

Suppose we have a degree-6 breakable vertex gadget with the desired behavior. Then to complete the reduction from Planar $(\{6\}, \emptyset)$-TRVB, we simply lay out the given multigraph in
wire for each edge from the gadget of one endpoint to the gadget of the other. Since finding a planar embedding for a graph is a polynomial time operation (provided one exists), such a reduction can be completed in polynomial time.

Below, we define more precisely what constraints a degree-6 vertex gadget must satisfy and then prove that the above reduction is correct. In short, the idea is that there is a correspondence between breaking a vertex and choosing the gadget solution shown in Figure 3-6c; under this correspondence the shape of the region inside the candidate set of edges is the same as the post-breaking multigraph. This region is connected and hole-free if and only if the post-breaking graph is connected and acyclic. Since a region has as its boundary a single cycle if and only if the region is connected and hole-free, we conclude that the candidate set of edges is a Hamiltonian cycle if and only if the answer to the corresponding Planar $(\{6\}, \emptyset)$-TRVB instance is "yes."

Suppose we start with an instance of Planar $(\{6\}, \emptyset)$-TRVB consisting of multigraph M and produce via the above reduction a grid graph G.

Definition 3.19. *Define a* candidate solution *for the Hamiltonian Cycle in Hexagonal Thin Grid Graphs instance G to be a set of edges C in G satisfying the following constraints: (1) every vertex in G is the endpoint of exactly two edges in C and (2) no one gadget or wire contains a cycle of edges from C entirely inside it.*

Failing to satisfy either of the constraints in the above definition is sufficient to disqualify a set of edges from being a Hamiltonian cycle. Thus the question of whether a Hamiltonian cycle exists in G is the same as the question of whether a candidate solution consisting of just one cycle exists.

Before, we described a degree-6 breakable vertex gadget by saying that the local constraints on a hypothetical Hamiltonian cycle allow only two possible solutions within the gadget. To make this more precise, intersecting the set of edges in the gadget with the set of edges in a candidate solution should only have two possible results. As before, the two possibilities are shown in Figure 3-6. We will refer to the posibilities in Figure 3-6b and Figure 3-6c as the connecting and disconnecting solutions respectively. We will refer to a vertex gadget using one of these solutions as a connecting or disconnecting vertex gadget.

Consider a wire between two gadgets and let C be any candidate solution. In both possible solutions of a gadget, there are two edges from the gadget entering into the wire. Thus two edges from C must enter the wire from each end. Simply applying the definition of a candidate solution (no cycles within a wire and every vertex must touch two edges), we can conclude that the edges in the wire that belong to C must be the boundary edges of the wire. For example, see Figure 3-7.

As a result, a candidate solution is completely constrained by the choice of behavior at each degree-6 vertex gadget. We can identify the disconnecting solution of a vertex gadget with the choice of breaking the corresponding vertex and identify the connecting solution with the choice of leaving the vertex unbroken. Under this correspondence we have a bijection between the choice of candidate solution in the Hamiltonian Cycle in Hexagonal Thin Grid Graphs instance G and the choice of what vertices to break in the Planar $(\{6\}, \emptyset)$-TRVB instance M. We show below that the candidate solutions which are Hamiltonian cycles

(a) (b)

Figure 3-7: The wire shown connects two degree-6 breakable vertex gadgets. The bold edges on the left are the edges that must belong to any candidate solution because of the behavior of the degree-6 vertex gadgets. The bold edges on the right show the resulting forced behavior in the wire.

correspond under this bijection with the choices of vertices to break whose breaking converts M into a tree. As a result, a Hamiltonian cycle exists in G if and only if it is possible to break vertices in M so as to obtain a tree, so we conclude that the reduction is correct.

By definition, a candidate solution C is a disjoint cycle cover of G (since every vertex has two edges in C incident on it). Therefore C seperates the plane into several regions. Among these regions, consider the ones which contain the interior of a wire. Let R be the union of these regions.

Looking at the possible behaviors of a candidate solution in a vertex gadget, it is easy to see that R will consist of the interior of each wire, one connected hole-free region from each connecting vertex gadget, and six connected hole-free regions from each disconnecting vertex gadget. In fact, the boundary of R is exactly the set of edges C.

Consider the above decomposition of R into sub-regions. The choice of candidate solution C corresponds with a set S of vertices to break in M. Let M_S be the version of M with the vertices of S broken. The sub-regions of R exactly correspond with the vertices and edges of M_S. If u and v are vertices in M then the edge (u, v) in M also occurrs in M_S and corresponds to the interior of the wire between the vertex gadgets for u and v. If $u \notin S$ is a vertex of M then u is also a vertex in M_S and corresponds to the sub-region of R from the vertex gadget for u. If $u \in S$ is a vertex of M then there are six vertices in M_S that result from the breaking of u, and these six vertices corresponds to the six sub-regions of R from the vertex gadget for u.

Clearly, the vertices and edges of M_S correspond to the sub-regions of R. Furthermore, it can be easily verified that two sub-regions of R touch if and only if those two sub-regions correspond to a vertex v and an edge e in M_S such that the v is an endpoint of e. Furthermore, there is no hole in R at such a point of contact between two sub-regions. We show below using these facts that M_S is connected if and only if R is connected and that M_S is acyclic if and only if R is hole-free.

M_S is connected if and only if there exists a path through M_S from any edge or vertex a to any other edge or vertex b. Such a path can be expressed as a list of edges and vertices $a = x_0, x_1, \ldots, x_k = b$ such that every consecutive pair x_i and x_{i+1} consists of one edge and one vertex that is an endpoint of that edge. We can create a corresponding list of sub-regions of R of the form $y_0, y_1, \ldots, y_k$ where y_i is the sub-region corresponding to edge or vertex x_i. In this list, every consecutive pair y_i and y_{i+1} consists of two touching subregions. Thus we see that M_S is connected if and only if for every pair of sub-regions a', b', there exists a list of the form $a' = y_0, y_1, \ldots, y_k = b'$ with y_i touching y_{i+1} for each i. Since every sub-region is itself connected, this last condition is equivalent to the condition that there exist a path in R between any two points; or in other words, M_S is connected if and only if R is connected.

Consider any cycle in M_S consisting of vertices and edges $x_0, x_1, \ldots, x_k = x_0$. This cycle corresponds to a cycle of sub-regions $y_0, y_1, \ldots, y_k = y_0$. Such a cycle of sub-regions will have an inner boundary, or in other words a hole. On the other hand, for any hole in R, we can list the subregions going around that hole: $y_0, y_1, \ldots, y_k = y_0$. Since each sub-region is hole free and each sub-region contact point does not have a hole, this list of subregions will correspond to a cycle of vertices and edges $x_0, x_1, \ldots, x_k = x_0$ in M_S. Thus we see that M_S is acyclic if and only if R is hole-free.

M_S is a tree, or in other words connected and acyclic, if and only if R is hole-free and connected. But R is hole-free and connected if and only if the boundary of R (which happens to be C) is exactly one cycle. Since C is a cycle cover of G, C is a Hamiltonian cycle if and only if C consists of exactly one cycle. Putting that all together, we see that M_S is a tree if and only if C is a Hamiltonian cycle in G.

As you can see, the constructed grid graph is Hamiltonian if and only if the input Planar $(\{6\}, \emptyset)$-TRVB instance is a "yes" instance, and therefore the reduction is correct.

3.4.2 Degree-6 breakable vertex gadget

Thus, all that is left is to demonstrate a degree-6 breakable vertex gadget of the desired form. The gadget we will use is shown in Figure 3-8a.

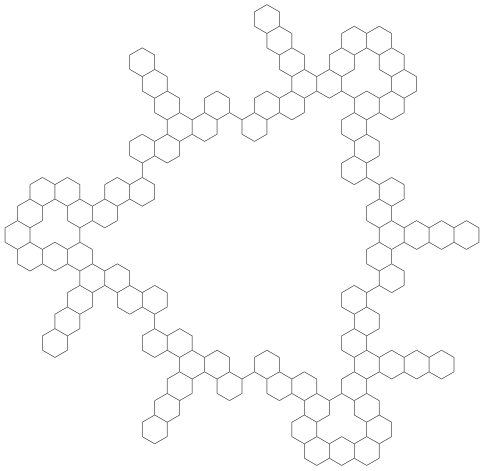
We show below that the gadget only has the two desired solutions.

Theorem 3.20. *The gadget shown in Figure 3-8a has exactly two possible solutions and they correspond with the solutions shown in Figure 3-6.*

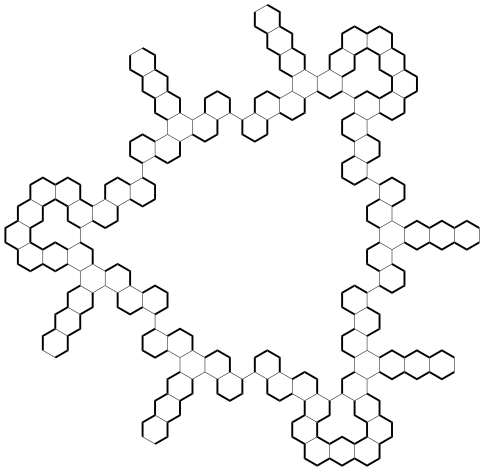
Proof. Figure 3-8b shows the gadget from Figure 3-8a, but with all edges that must be in any candidate solution bold. These constraints can be derived purely from the fact that every vertex must be the endpoint of exactly two edges in a candidate solution.

The gadget contains six contiguous regions of pixels, arranged in a cycle with a single edge that is not part of any pixel between adjacent regions. We claim that either all of these single edges must be in the candidate solution or none of them. It can be verified that these two possibilities lead, again via the constraints on candidate solutions, to the two solutions shown in Figures 3-8c and 3-8d.

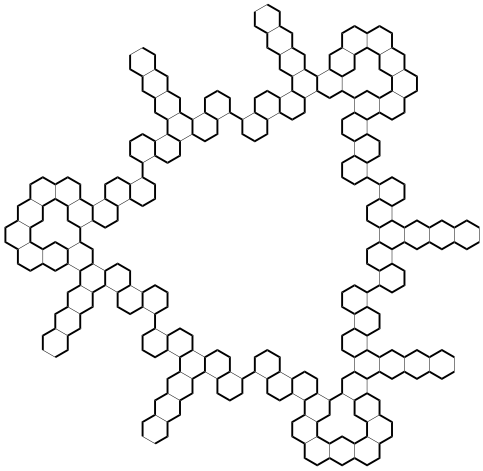
All that's left to show is that our claim is correct: that using only some of the single edges in a candidate solution is impossible. Suppose for the sake of contradiction that a candidate solution exists which uses some but not all of these edges. Going around the cycle of regions, there will be some region where we transition from using the single edge to not using it. That region will have exactly one of its two single edges in the candidate solution. Thus, the candidate solution will enter or exit the region exactly three times: twice through the wire that exists the gadget from this region and once through the single edge. Since this is impossible for a cycle cover, we arrive at the desired contradiction. □



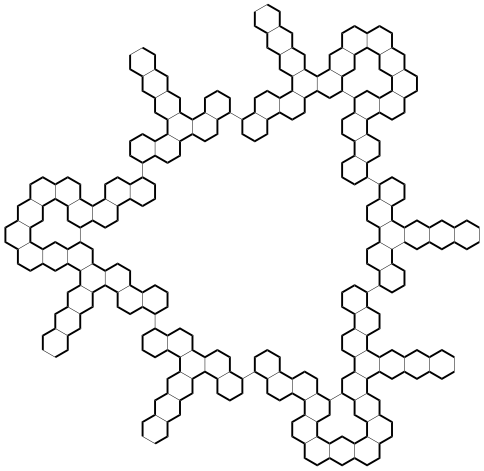
(a) A degree-6 breakable vertex gadget (with six wires).



(b) The gadget with edges that must be in a Hamiltonian cycle in bold.



(c) The first solution of the gadget with edges chosen for the Hamiltonian cycle bold. Note that the regions inside the wires are connected via this gadget.



(d) The second solution of the gadget with edges chosen for the Hamiltonian cycle bold. Note that the regions inside the wires are disconnected via this gadget.

Figure 3-8: A degree-6 breakable vertex gadget together with the two possible solutions.

3.5 Hamiltonian Cycle in Square Polygonal Grid Graphs is NP-complete

In this section, we show that the Hamiltonian Cycle in Square Polygonal Grid Graphs problem is NP-complete. Membership in NP is trivial, while NP-hardness follows from a polynomial-time reduction. We reduce from the Planar Monotone Rectilinear 3SAT problem—which was shown NP-hard in [5]—to the Hamiltonian Cycle in Square Polygonal Grid Graphs problem. In order to state the Planar Monotone Rectilinear 3SAT problem, we first need some preliminary terms:

Definition 3.21. *A* rectilinear embedding *of a CNF formula into a plane is a planar embedding of the variable-clause bipartite graph for the CNF formula into the plane such that the vertices (variables and clauses) are mapped to horizontal segments, the edges are mapped to vertical segments, and the variables all lie on the x axis. Furthermore, a* monotone rectilinear embedding *of a CNF formula into a plane is a rectilinear embedding with the additional property that the clauses positioned above the x axis contain only positive literals while clauses positioned below the x axis contain only negative literals.*

Figure 3-9 shows an example of a rectilinear embedding. This embedding could represent several possible CNF formulas, such as $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_4} \vee x_5) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_5})$ or $(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_3} \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_5)$. If, however, this is a monotone rectilinear embedding then a specific CNF formula, $(x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_5})$ is implied.

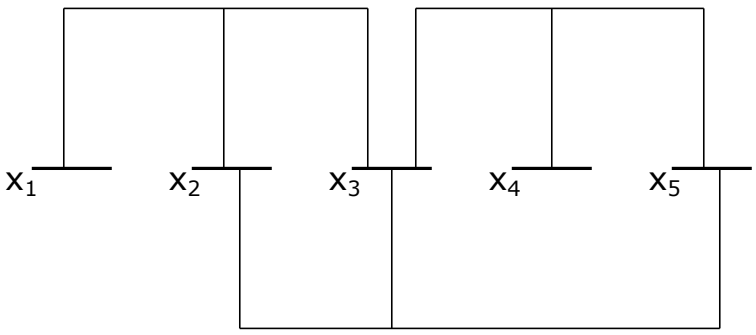


Figure 3-9: A rectilinear embedding

Now the problem we are reducing from can be stated as follows:

Problem 3.22. *The problem Planar Monotone Rectilinear 3SAT decides, for every monotone rectilinear embedding of a 3-CNF formula into the plane, whether the given instance is satisfiable.*

Our overall strategy for reducing Planar Monotone Rectilinear 3SAT to Hamiltonian Cycle in Square Polygonal Grid Graphs is the following:

1. We begin by describing several simple gadgets.
   (a) A wire gadget consists of a path of pixels.
   (b) A one-enforcer gadget, when inserted into a wire, enforces the fact that any Hamiltonian cycle in the grid graph must pass through the wire only once, zig-zagging through the wire in one direction.

(c) A two-enforcer gadget, when inserted into a wire, enforces the fact that any Hamiltonian cycle in the grid graph must pass through the wire twice, once in each direction.

2. We can then combine these gadgets to form variable and clause gadgets, which we will use to simulate a Planar Monotone Rectilinear 3SAT instance.

   (a) A variable gadget consists of a loop of wire, which when correctly inserted into a one-enforced wire enforces certain constraints on any Hamiltonian cycle (provided one exists). In particular, among the two wires in the gadget (the top and bottom halves of the loop of wire), one will be one-enforced, and the other will be two-enforced. This choice corresponds to a choice of value for the variable.

   (b) A clause gadget attaches to three variable gadgets (attaching to either the top or bottom wires of each) in such a way that the previously stated facts about variable gadgets continue to hold. If all three attaching points connect the clause gadget to one-enforced wire (due to the choices at the variable gadgets) then there exists no Hamiltonian cycle in the grid graph. Conversely, whenever this fails to occur, it is possible to modify one of the pieces-of-cycle passing through the variable gadgets to also pass through every point in the clause gadget.

3. Next, we assemble a grid graph out of these gadgets to simulate a Planar Monotone Rectilinear 3SAT instance. We create a variable gadget for every variable and connect them in a loop with one-enforcers appropriately placed. We then add clause gadgets outside the loop and inside the loop to represent the positive and negative clauses. For an example, see Figure 3-10.

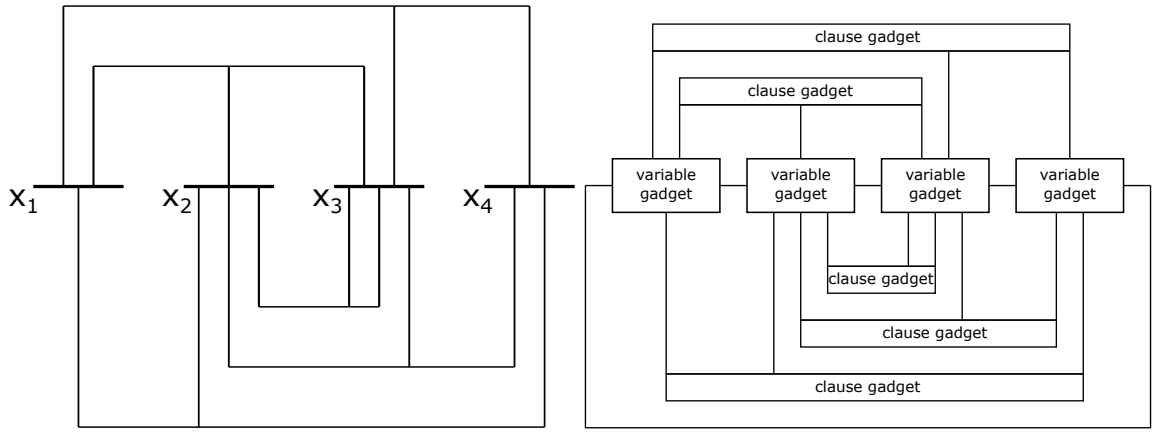
4. If there exists a satisfying assignment then we can construct a cycle passing through the one-enforced loop and through the variable gadgets in such a way that each clause gadget attaches to at least one two-enforced wire (among the three variable gadget wires that it attaches to). Then by the clause gadget properties listed above, we can modify that cycle to also pass through every point in each clause gadget. This yields a Hamiltonian cycle. If there exists no satisfying assignment then any cycle passing through the one-enforced wire loop and variable gadgets will result in at least one clause gadget being attached to three one-enforced wires. Then by the clause gadget properties listed above, no Hamiltonian cycle exists. This shows that the reduction is answer preserving.

We follow this outline below to prove that

Theorem 3.23. *There exists a polynomial time reduction from the Planar Monotone Rectilinear 3SAT problem to the Hamiltonian Cycle in Square Polygonal Grid Graphs problem.*

and therefore that

Corollary 3.24. *The Hamiltonian Cycle in Square Polygonal Grid Graphs problem is NP-complete.*



(a) The Planar Monotone Rectilinear 3SAT instance $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$

(b) The corresponding Hamiltonian Cycle in Square Polygonal Grid Graphs instance

Figure 3-10: An example of the reduction from Planar Monotone Rectilinear 3SAT to Hamiltonian Cycle in Square Polygonal Grid Graphs

3.5.1 Simple gadgets

Wires

The simplest gadget we will use will be the wire, which is simply a path of pixels. As shown in Figure 3-11, there are multiple ways to pass a cycle through a wire. The important distinction to make is between "two-enforced" wires for which the cycle passes through most pixels twice, once in each direction (the first two solutions in Figure 3-11), and "one-enforced" wires for which the cycle passes through most pixels once (the final two solutions in Figure 3-11).

(a) Gadget

(b) Solutions

Figure 3-11: A wire

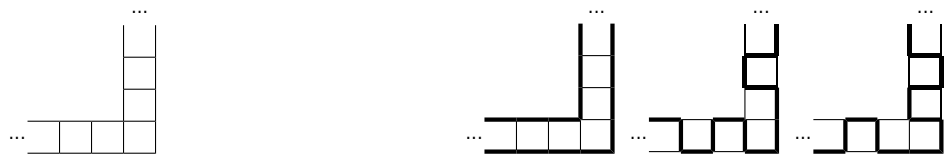
Note that the distinction between one-enforced and two-enforced wires must propagate through a wire by a simple contradiction argument: if a cycle goes into a wire once from one side and comes out twice from the other then it enters that wire an odd number of times in all, which is impossible for a cycle.

Furthermore, as seen in Figure 3-12, even if the wire turns (provided the turns are not too close), it is possible to propagate this property.

One-enforcers

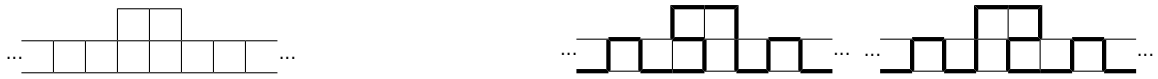
A one-enforcer, when inserted into a wire causes the wire to be one-enforced in a particular parity. The gadget is shown in Figure 3-13a. The two possible solutions are shown in Figure 3-13b.

In both solutions, the behavior of the cycle in the wire attached to the gadget is the same: a one-enforced wire of a particular parity. The parity is different on each side; in



(a) Gadget (b) Solutions

Figure 3-12: A turning wire



(a) Gadget (b) Solutions

Figure 3-13: A one-enforcer gadget

other words, the gadget causes a parity shift in addition to its other effects.

Two-enforcers

Inserting a two-enforcer into a wire causes that wire to be two-enforced. The shape of a two enforcer is shown in Figure 3-14a. Every edge adjacent to a vertex with only two neighbors must be in a Hamiltonian cycle if one exists. Thus all the bold edges in Figure 3-14b must be in the cycle. But then the section of gadget in the middle is a two-enforced wire and this property propagates along the wire in both directions away from the gadget.

(a) Gadget (b) Solutions

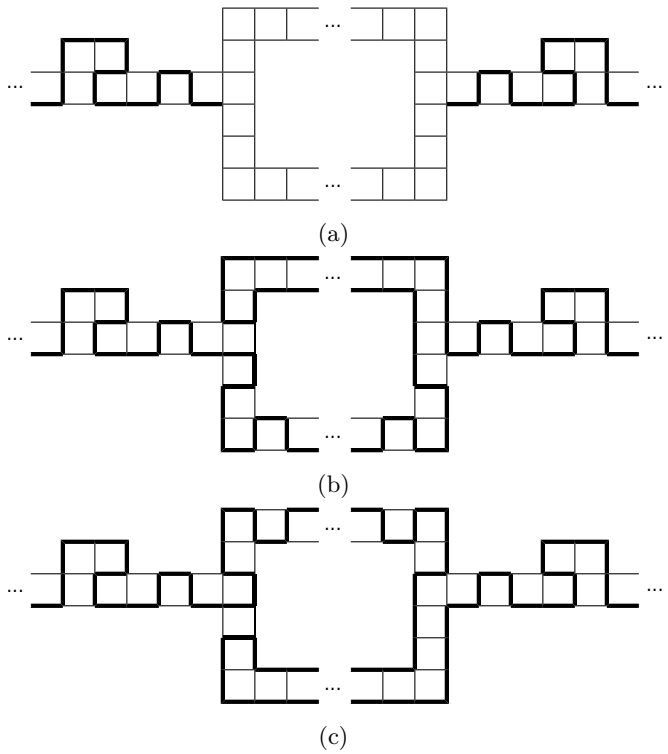
Figure 3-14: A two-enforcer gadget

3.5.2 Variable gadget

A variable gadget consists of a loop of wire of odd width and height 6 that is inserted into a one-enforced wire of a particular parity. Figure 3-15a shows a variable gadget together with the appropriate one-enforcers necessary for the gadget to properly function.

Consider just one half of the gadget (Figure 3-16a). The possible ways for a cycle to pass through this gadget half are listed in Figure 3-16b. The important thing to note is that in all cases one of the two branches is one-enforced and the other is two-enforced.

Putting the variable gadget together from two halves, we can figure out the legal solutions by assigning solutions to the two halves in ways that don't result in isolated cycles. The only two possible ways (up to reflection) for a cycle to pass through a variable gadget are shown in Figures 3-15b and 3-15c. As you can see, the variable gadget allows a choice: either the top or bottom wire must be two-enforced while the other must be one-enforced.

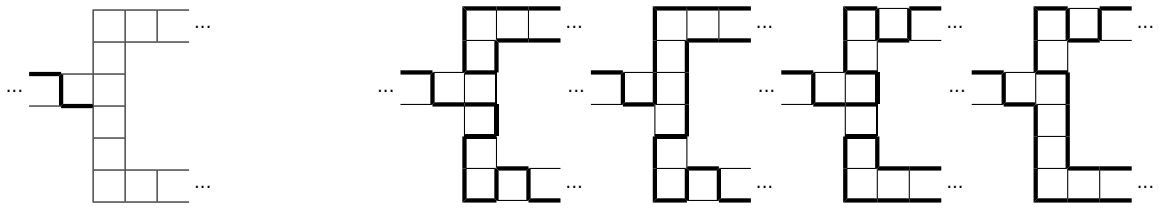


(a)

(b)

(c)

Figure 3-15: A variable gadget (Figure 3-15a) and two ways a cycle can pass through the variable gadget (Figures 3-15b and 3-15c)



(a) Gadget

(b) Solutions

Figure 3-16: One half of a variable gadget

3.5.3 Clause gadget

A clause gadget is a long horizontal wire with three two-enforced vertical wires brought vertically down (or up) from the horizontal wire. It is shown in Figure 3-17 with the two-enforcers omitted but with bold edges as appropriate to indicate that the wires are two-enforced.

Figure 3-17: Two clause gadgets

When used, this gadget will attach each of the three two-enforced vertical wires to a variable gadget. A variable gadget has a top horizontal wire and a bottom horizontal wire. The vertical two-enforced wire from the clause is connected directly (in a particular place described later) to one of these two wires (the bottom one if the clause's vertical wires go up from the horizontal clause section and the top if the clause's vertical wires go down from the horizontal clause section). This is a modification to the variable gadget, and so we must verify that the variable gadget continues to work as expected (and no other way) with this addition.

In fact, a rigorous derivation of the behavior of a cycle in a variable gadget relies on a repeated application of the same parity argument. For example, to show that the two wires in a variable half cannot be both one-enforced or both two-enforced, we would argue that the total number of arcs of cycle at the edge of that gadget must be even, and therefore cannot be 3 or 5 (which would occur if the wires were both one- or two-enforced). Similarly, to show that the two-enforced wire from one half of the variable gadget cannot coincide with the one-enforced wire from the other, we argue based on the parity of the number of cycle arcs entering the wire. Since our modification only attaches a two-enforced wire, and since that addition only ever adds an even number of cycle arcs to a region, we see that the same arguments must continue to hold and therefore that the set of possible ways for a cycle to pass through a vertex gadget remains the same (one variable wire will be two-enforced while the other will be one-enforced).

So consider the horizontal variable wire that the clause gadget connects to. That wire could be either one-enforced or two-enforced. Furthermore, if the wire is one-enforced, the cycle follows a particular parity of zigzag that is known in advance. When attaching a clause gadget to the horizontal variable wire, the vertical clause wire is aligned so that if the variable wire is one-enforced, the two wires match up as shown in Figure 3-18.

We prove that if the variable wire is one-enforced, the situation from Figure 3-18 is exactly what will occur. Without loss of generality, suppose the horizontal wire is the top wire of the variable node. The cycle in the one-enforced variable wire propagates from the left through this wire. And the two arcs of the cycle in the two-enforced vertical clause wire propagate down. At this point, all we know is what is shown in Figure 3-19a. Consider the circled node in that figure. It must contribute the next two edges as shown in Figure 3-19b. Then the next circled node contributes the next edge as shown in Figure 3-19c. Finally, the circled vertex in that figure contributes two edges to the cycle in the horizontal variable wire as shown in Figure 3-19d. Thus the horizontal variable wire continues to be one-enforced

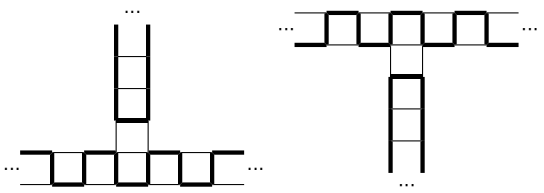


Figure 3-18: The junction between a variable gadget's horizontal wire and one of the three vertical wires from a clause gadgets; the case shown is where the variable gadget's wire is one-enforced; in this case, the cycle does not "match up"

despite the fact that the vertical wire is attached. As desired, the cycle must pass through this clause/variable junction as shown in Figure 3-18.

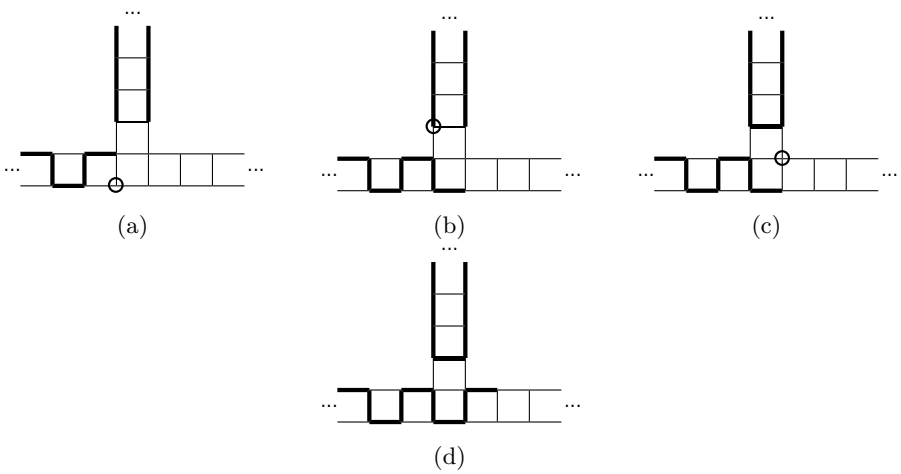


Figure 3-19: The parts of the cycle in the variable and clause wires do not connect if the variable wire is one-enforced

On the other hand, when the horizontal variable wire is two-enforced, the parts of the cycle inside the variable and clause gadgets are free to connect as shown in Figure 3-20. Note however, that the parts of the cycle in the two gadgets are also free to not connect, which is useful when multiple variables satisfy a clause.

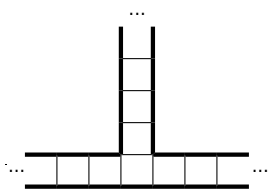


Figure 3-20: The parts of the cycle in the variable and clause wires can connect if the variable wire is two-enforced

Thus we can conclude the following two facts about our clause gadget.

If a Hamiltonian cycle passes through each vertex gadget such that some clause gadget attaches to some three one-enforced wires, then the part of the cycle in the clause gadget has no way to connect to the part in the variable gadgets and so we don't actually have a Hamiltonian cycle (rather we have a contradiction). In other words, a Hamiltonian

cycle passing through each vertex gadget must attach each clause gadget to at least one two-enforced horizontal variable wire.

If a grid graph has a Hamiltonian cycle and we add a clause gadget, attaching it to three horizontal vertex wires of which at least one is two-enforced, then the new graph is also Hamiltonian. This is because we can simply join the cycle that forms the boundary of the clause gadget to the pre-existing Hamiltonian cycle at one of the two-enforced horizontal variable wires (as in Figure 3-20), merging them into a Hamiltonian cycle for the entire graph.

3.5.4 Overall reduction

Suppose we are given an instance of Planar Monotone Rectilinear 3SAT, or in other words a monotone rectilinear embedding of a 3-CNF formula in the plane. We convert this instance into a grid graph as follows. Each variable segment gets replaced by a variable gadget, which are connected in order with wires and one-enforcers. Then a long wire (possibly with extra one-enforcers for parity) connects the two variable gadgets at the ends, forming a big loop of variable gadgets (and one-enforcers). After that, we add clause gadgets for each clause. The positive clauses get a clause gadget above the variable gadgets while the negative clauses get a clause gadget inside the loop. An example schematic of what the final result might be is shown in Figure 3-21, corresponding to the Planar Monotone Rectilinear 3SAT instance shown in Figure 3-10.

As you can see, the resulting instance of Hamiltonian Cycle in Square Polygonal Grid Graphs (the resulting grid graph) is very similar in shape to the original embedding. This similarity in shape between the embedding inputted into the reduction and the resulting instance of Hamiltonian Cycle in Square Polygonal Grid Graphs can be used as an argument that this is a polynomial time reduction. It is very easy to construct the Hamiltonian Cycle in Square Polygonal Grid Graphs instance from the embedding, and furthermore, the resulting graph is barely larger than the embedding. Due to the simplicity of the reduction, it is possible to construct the grid graph in time proportional to the size of the grid graph, but that size is itself linear in the size of the embedding. Thus we see that the reduction given above runs in polynomial time.

The remaining goal is to show that the reduction is answer preserving.

Consider first the case that the grid graph is Hamiltonian. Then we construct a variable assignment by assigning the value true to a variable if and only if that variable's variable gadget has the top wire two-enforced. Consider any [positive/negative] clause $[(x_i \vee x_j \vee x_k)/(\bar{x_i} \vee \bar{x_j} \vee \bar{x_k})]$ under this assignment. The corresponding gadget for a [positive/negative] clause is [outside/inside] the large loop in the grid graph; thus the clause gadget is attached to the [top/bottom] wires of the variable gadgets for variables x_i, x_j, and x_k. By one of our derived properties for clause gadgets, we know that each clause is attached to at least one two-enforced wire; thus the [top/bottom] wire for the x_i, x_j, or x_k variable gadget must be two-enforced. Equivalently, either x_i, x_j, or x_k must be [true/false]. Thus, at least one variable in the clause is [true/false], and so at least one literal in the clause is true. In other words, the entire clause must be true. Then we see that we have found a satisfying assignment.

Next consider the case that a satisfying assignment for the formula exists. Consider the grid graph with all the clause gadgets removed (just a loop of variable gadgets and one-enforcers). Certainly a Hamiltonian cycle exists in this graph; in fact, many do: the cycle has two choices of behavior at each variable gadget. Construct the particular Hamiltonian cycle

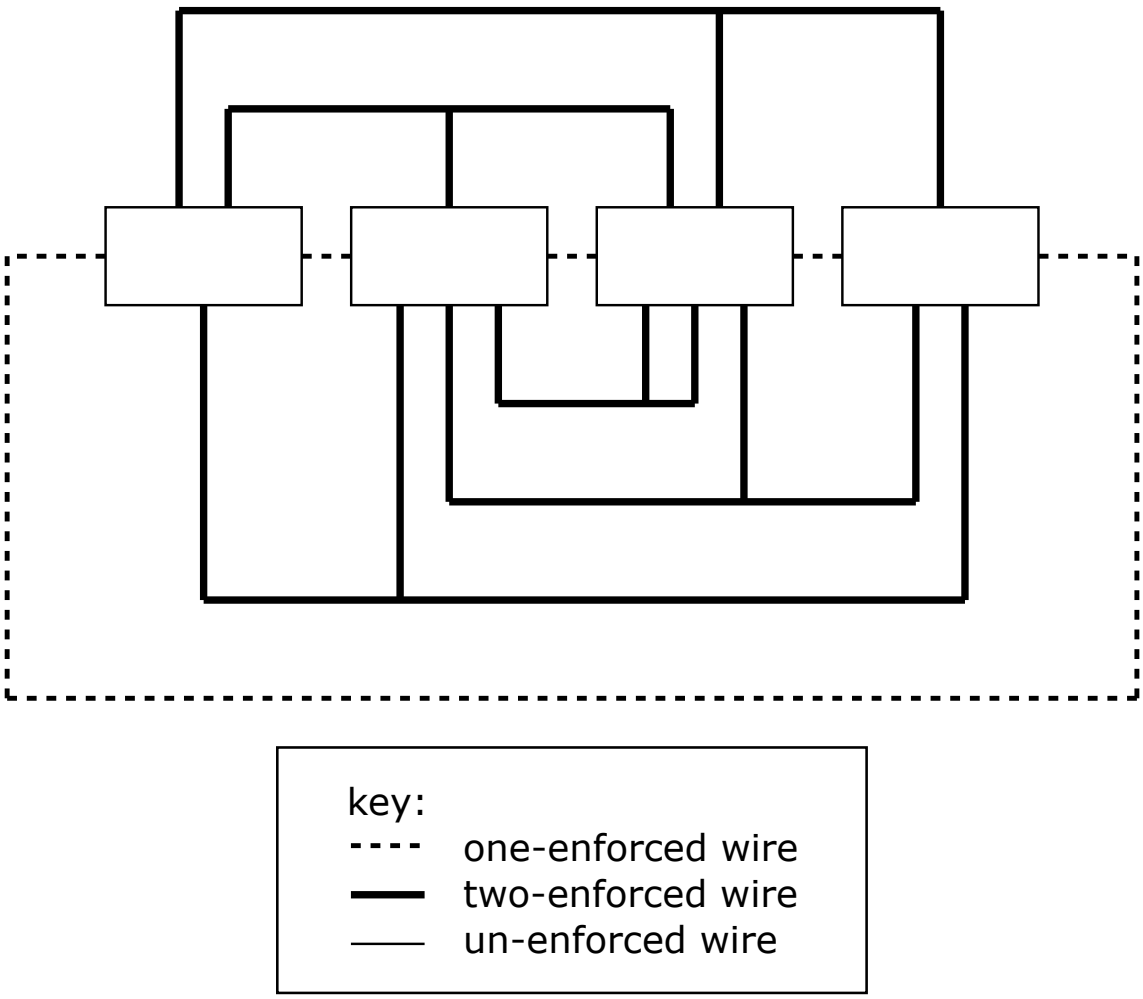


Figure 3-21: An example schematic for the resulting grid graph that could be produced from the instance of Planar Monotone Rectilinear 3SAT in Figure 3-10

in which a variable gadget's top wire is two-enforced if and only if the variable is assigned a value of true. We will add the clause gadgets back into the graph one at a time. By one of our derived properties for clause gadgets, we know that if the clause being added attaches to at least one two-enforced wire then we can extend the Hamiltonian cycle to the new graph. Consider a clause gadget for a [positive/negative] clause $[(x_i \vee x_j \vee x_k)/(\bar{x_i} \vee \bar{x_j} \vee \bar{x_k})]$. The gadget is [outside/inside] the large loop in the grid graph and is therefore attached to the [top/bottom] wires of the variable gadgets for variables x_i, x_j, and x_k. But since the clause is satisfied, either x_i, x_j, or x_k must be [true/false] and so the [top/bottom] wire for that variable gadget must be two-enforced. Thus the clause attaches to at least one two-enforced wire. Thus, as we add the clause gadgets back into the graph one at a time, the graph remains Hamiltonian. Clearly then the full graph is Hamiltonian.

As desired, we see that the grid graph in question is Hamiltonian if and only if the 3-CNF formula is satisfiable and that therefore we have described a polynomial-time answer-preserving reduction.

3.6 Conclusion and further work

We showed that Hamiltonian Cycle in Thin Polygonal Grid Graphs is solvable in polynomial time for every shape of grid graph. In addition, we showed that Hamiltonian Cycle in Square Polygonal Grid Graphs and Hamiltonian Cycle in Hexagonal Thin Grid Graphs are both NP-complete. Having determined this, we have proved the hardness of two of the problems left open by Arkin et al. [3], as well as determining the complexity of three problems not addressed in that paper.

This leaves only one of Arkin et al.'s open problems, namely, the complexity of Hamiltonian Cycle in Hexagonal Solid Grid Graphs. Arkin et al. conjectured that this problem can be solved in polynomial time, based on the idea that the polynomial-time algorithm for Hamiltonian Cycle in Square Solid Grid Graphs could be adapted to also solve Hamiltonian Cycle in Hexagonal Solid Grid Graphs. The algorithm is a cycle-merging algorithm which starts with a 2-factor for the grid graph and progressively merges cycles until the 2-factor has one component (a Hamiltonian cycle) or until further merging is impossible. Many of the ideas appear relevant to hexagonal grid graphs as well, but a direct translation of the algorithm from solid square grid graphs and the relevant correctness proofs to hexagonal grid graphs fails. In order to make this approach work, some new insight seems necessary. Nevertheless, having struggled with this problem for some time, we believe the conjecture to be correct: Hamiltonian Cycle in Hexagonal Solid Grid Graphs can probably be solved in polynomial time.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Solving the Rubik's Cube Optimally is NP-complete

4.1 Introduction

The overall purpose of this chapter is to address the computational difficulty of optimally solving Rubik's Cubes. In particular, consider the decision problem which asks for a given puzzle configuration whether that puzzle can be solved in a given number of moves. We show that this problem is NP-complete for several different types of puzzles: for the generalized $n \times n$ Rubik's Square (a simplified version of the Rubik's Cube) and for the generalized $n \times n \times n$ Rubik's Cube under two different move models. These results close a problem that has been repeatedly posed as far back as 1984 [6, 20, 23] and has until now remained open [15].

In Section 4.2, we formally introduce the decision problems regarding Rubik's Squares and Rubik's Cubes whose complexity we will analyze. Then in Section 4.3, we introduce the variant of the Hamiltonicity problem that we will reduce from—Promise Cubical Hamiltonian Path—and prove this problem to be NP-hard. Next, we prove that the problems regarding the Rubik's Square are NP-complete in Section 4.4 by reducing from Promise Cubical Hamiltonian Path. After that, we apply the same ideas in Section 4.5 to a more complicated proof of NP-hardness for the problems regarding the Rubik's Cube. Finally, we discuss possible next steps in Section 4.6.

The research in this chapter is joint work with Erik Demaine and Sarah Eisenstat.

4.2 Rubik's Cube and Rubik's Square problems

4.2.1 Rubik's Square

We begin with a simpler model based on the Rubik's Cube which we will refer to as the Rubik's Square. In this model, a puzzle consists of an $n \times n$ array of unit cubes, called *cubies* to avoid ambiguity. Every cubie face on the outside of the puzzle has a colored (red, blue, green, white, yellow, or orange) sticker. The goal of the puzzle is to use a sequence of moves to rearrange the cubies such that each face of the puzzle is monochromatic in a different color. A *move* consists of flipping a single row or column in the array through space via a rotation in the long direction as demonstrated in Figure 4-1.

We are concerned with the following decision problem:

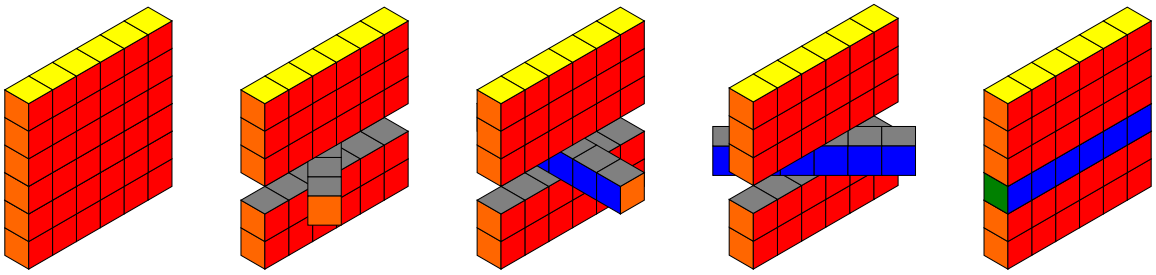


Figure 4-1: A single move in an example 6×6 Rubik's Square.

Problem 4.1. *The **Rubik's Square** problem has as input an $n \times n$ Rubik's Square configuration and a value k. The goal is to decide whether a Rubik's Square in configuration C can be solved in k moves or fewer.*

This type of puzzle was previously introduced in [11] as the $n \times n \times 1$ Rubik's Cube. In that paper, the authors showed that deciding whether it is possible to solve the $n \times n \times 1$ Rubik's Cube in a given number of moves is NP-complete when the puzzle is allowed to have missing stickers (where the puzzle is considered solved if each face contains stickers of only one color).

4.2.2 Rubik's Cube

Next consider the Rubik's Cube puzzle. An $n \times n \times n$ Rubik's Cube is a cube consisting of n^3 unit cubes called *cubies*. Every face of a cubie that is on the exterior of the cube has a colored (red, blue, green, white, yellow, or orange) sticker. The goal of the puzzle is to use a sequence of moves to reconfigure the cubies in such a way that each face of the cube ends up monochromatic in a different color. A *move count metric* is a convention for counting moves in a Rubik's Cube. Several common move count metrics for Rubik's Cubes are listed in [25]. As discussed in [7], however, many common move count metrics do not easily generalize to $n > 3$ or are not of any theoretical interest. In this chapter, we will restrict our attention to two move count metrics called the Slice Turn Metric and the Slice Quarter Turn Metric. Both of these metrics use the same type of motion to define a move. Consider the subdivision of the Rubik's Cube's volume into n *slices* of dimension $1 \times n \times n$ (or $n \times 1 \times n$ or $n \times n \times 1$). In the Slice Turn Metric (STM), a *move* is a rotation of a single slice by any multiple of 90°. Similarly, in the Slice Quarter Turn Metric (SQTM), a *move* is a rotation of a single slice by an angle of 90° in either direction. An example SQTM move is shown in Figure 4-2.

We are concerned with the following decision problems:

Problem 4.2. *The **STM/SQTM Rubik's Cube problem** takes as input a configuration of a Rubik's Cube together with a number k. The goal is to decide whether a Rubik's Cube in configuration C can be solved in k STM/SQTM moves.*

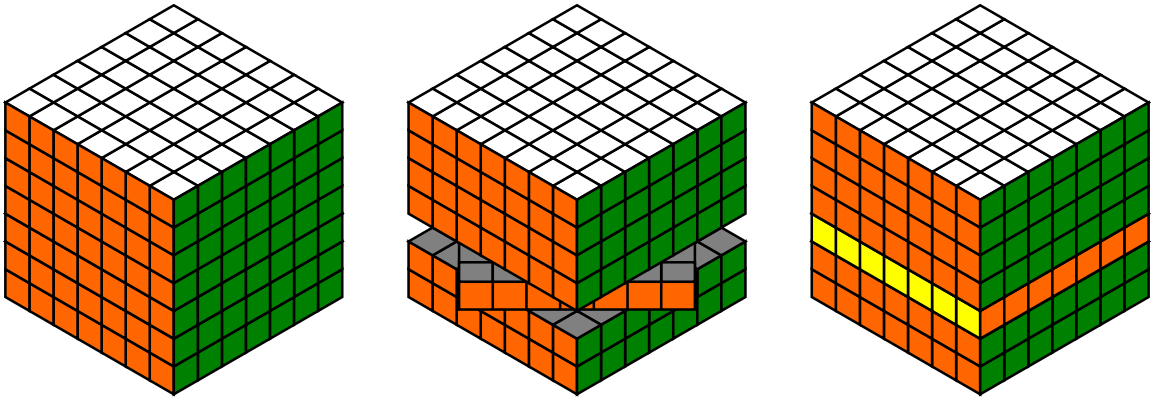


Figure 4-2: A single slice rotation in an example $7 \times 7 \times 7$ Rubik's Cube.

4.2.3 Notation

Next we define some notation for dealing with the Rubik's Cube and Rubik's Square problems.

To begin, we need a way to refer to cubies and stickers. For this purpose, we orient the puzzle to be axis-aligned. In the case of the Rubik's Square we arrange the $n \times n$ array of cubies in the x and y directions and we refer to a cubie by stating its x and y coordinates. In the case of the Rubik's Cube, we refer to a cubie by stating its x, y, and z coordinates. To refer to a sticker in either puzzle, we need only specify the face on which that sticker resides (e.g. "top" or "$+z$") and also the two coordinates of the sticker along the surface of the face (e.g. the x and y coordinates for a sticker on the $+z$ face).

If $n = 2a+1$ is odd, then we will let the coordinates of the cubies in each direction range over the set $\{-a, -(a-1), \ldots, -1, 0, 1, \ldots, a-1, a\}$. This is equivalent to centering the puzzle at the origin. If, however, $n = 2a$ is even, then we let the coordinates of the cubies in each direction range over the set $\{-a, -(a-1), \ldots, -1\} \cup \{1, \ldots, a-1, a\}$. In this case, the coordinate scheme does not correspond with a standard coordinate sheme no matter how we translate the cube. This coordinate scheme is a good idea for the following reason: under this scheme, if a move relocates a sticker, the coordinates of that sticker remain the same up to permutation and negation.

Next, we need a way to distinguish the sets of cubies affected by a move from each other.

In the Rubik's Square, there are two types of moves. The first type of move, which we will call a *row move* or a *y move*, affects all the cubies with some particular y coordinate. The second type of move, which we will call a *column move* or an *x move* affects all the cubies with some particular x coordinate. We will refer to the set of cubies affected by a row move as a *row* and refer to the set of cubies affected by a column move as a *column*. In order to identify a move, we must identify which row or column is being flipped, by specifying whether the move is a row or column move as well as the index of the coordinate shared by all the moved cubies (e.g. the index -5 row move is the move that affects the cubies with $y = -5$).

In the Rubik's Cube, each STM/SQTM move affects a single slice of n^2 cubies sharing some coordinate. If the cubies share an x (or y or z) coordinate, then we call the slice an *x (or y or z) slice*. As with the Rubik's Square, we identify the slice by its normal direction together with its cubies' index in that direction (e.g. the $x = 3$ slice). We will also refer to the six slices at the boundaries of the Cube as *face slices* (e.g. the $+x$ face slice).

A move in a Rubik's Cube can be named by identifying the slice being rotated and the amount of rotation. We split this up into the following five pieces of information: the normal direction to the slice, the sign of the index of the slice, the absolute value of the index of the slice, the amount of rotation, and the direction of rotation. Splitting the information up in this way allows us not only to refer to individual moves (by specifying all five pieces of information) but also to refer to interesting sets of moves (by omitting one or more of the pieces of information).

To identify the normal direction to a slice, we simply specify x, y, or z; for example, we could refer to a move as an x move whenever the rotating slice is normal to the x direction. We will use two methods to identify the sign of the index of a moved slice. Sometimes we will refer to positive moves or negative moves, and sometimes we will combine this information with the normal direction and specify that the move is a $+x$, $-x$, $+y$, $-y$, $+z$, or $-z$ move. We use the term *index-v move* to refer to a move rotating a slice whose index has absolute value v. In the particular case that the slice rotated is a face slice, we instead use the term *face move*. We refer to a move as a *turn* if the angle of rotation is 90° and as a *flip* if the angle of rotation is 180°. In the case that the angle of rotation is 90°, we can specify further by using the terms *clockwise turn* and *counterclockwise turn*. We make the notational convention that clockwise and counterclockwise rotations around the x, y, or z axes are labeled according to the direction of rotation when looking from the direction of positive x, y, or z.

We also extend the same naming conventions to the Rubik's Square moves. For example, a positive row move is any row move with positive index and an index-v move is any move with index $\pm v$.

4.2.4 Group-theoretic approach

An alternative way to look at the Rubik's Square and Rubik's Cube problems is through the lens of group theory. The transformations that can be applied to a Rubik's Square or Rubik's Cube by a sequence of moves form a group with composition as the group operation. Define RS_n to be the group of possible sticker permutations in an $n \times n$ Rubik's Square and define RC_n to be the group of possible sticker permutations in an $n \times n \times n$ Rubik's Cube.

Consider the moves possible in an $n \times n$ Rubik's Square or an $n \times n \times n$ Rubik's Cube. Each such move has a corresponding element in group RS_n or RC_n.

For the Rubik's Square, let $x_i \in RS_n$ be the transformation of flipping the column with index i in an $n \times n$ Rubik's Square and let y_i be the transformation of flipping the row with index i in the Square. Then if I is the set of row/column indices in an $n \times n$ Rubik's Square we have that RS_n is generated by the set of group elements $\bigcup_{i \in I} \{x_i, y_i\}$.

Similarly, for the Rubik's Cube, let x_i, y_i, and z_i in RC_n be the transformations corresponding to clockwise turns of x, y, or z slices with index i. Then if I is the set of row/column indices in an $n \times n \times n$ Rubik's Cube we have that RC_n is generated by the set of group elements $\bigcup_{i \in I} \{x_i, y_i, z_i\}$.

Using these groups we obtain a new way of identifying puzzle configurations. Let C_0 be a canonical solved configuration of a Rubik's Square or Rubik's Cube puzzle. For the $n \times n$ Rubik's Square, define C_0 to have top face red, bottom face blue, and the other four faces green, orange, yellow, and white in some fixed order. For the $n \times n \times n$ Rubik's Cube, let C_0 have the following face colors: the $+x$ face is orange, the $-x$ face is red, the $+y$ face is green, the $-y$ face is yellow, the $+z$ face is white, and the $-z$ face is blue. Then from any element of RS_n or RC_n, we can construct a configuration of the corresponding puzzle

by applying that element to C_0. In other words, every transformation $t \in RS_n$ or $t \in RC_n$ corresponds with the configuration $C_t = t(C_0)$ of the $n \times n$ Rubik's Square or $n \times n \times n$ Rubik's Cube that is obtained by applying t to C_0.

Using this idea, we define a new series of problems:

Problem 4.3. *The **Group Rubik's Square** problem has as input a transformation $t \in RS_n$ and a value k. The goal is to decide whether the transformation t can be reversed by a sequence of at most k transformations corresponding to Rubik's Square moves. In other words, the answer is "yes" if and only if the transformation t can be reversed by a sequence of at most k transformations of the form x_i or y_i.*

Problem 4.4. *The **Group STM/SQTM Rubik's Cube** problem has as input a transformation $t \in RC_n$ and a value k. The goal is to decide whether the transformation t can be reversed by a sequence of at most k transformations corresponding with legal Rubik's Cube moves under move count metric STM/SQTM.*

We can interpret these problems as variants of the Rubik's Square or Rubik's Cube problems. For example, the Rubik's Square problem asks whether it is possible (in a given number of moves) to unscramble a Rubik's Square configuration so that each face ends up monochromatic, while the Group Rubik's Square problem asks whether it is possible (in a given number of moves) to unscramble a Rubik's Square configuration so that each sticker goes back to its exact position in the originally solved configuration C_0. As you see, the Group Rubik's Square problem, as a puzzle, is just a more difficult variant of the puzzle: instead of asking the player to move all the stickers of the same color to the same face, this variant asks the player to move each stickers to the exact correct position. Similarly, the Group STM/SQTM Rubik's Cube problem as a puzzle asks the player to move each sticker to an exact position. These problems can have practical applications with physical puzzles. For example, some Rubik's Cubes have pictures split up over the stickers of each face instead of just monochromatic colors on the stickers. For these puzzles, as long as no two stickers are the same, the Group STM/SQTM Rubik's Cube problem is more applicable than the STM/SQTM Rubik's Cube problem (which can leave a face "monochromatic" but scrambled in image).

We formalize the idea that the Group version of the puzzle is a strictly more difficult puzzle in the following lemmas:

Lemma 4.5. *If (t, k) is a "yes" instance to the Group Rubik's Square problem, then $(t(C_0), k)$ is a "yes" instance to the Rubik's Square problem.*

Lemma 4.6. *If (t, k) is a "yes" instance to the Group STM/SQTM Rubik's Cube problem, then $(t(C_0), k)$ is a "yes" instance to the STM/SQTM Rubik's Cube problem.*

The proof of each of these lemmas is the same. If (t, k) is a "yes" instance to the Group variants of the puzzle problems, then t can be inverted using at most k elements corresponding to moves. Applying exactly those moves to $t(C_0)$ yields configuration C_0, which is a solved configuration of the cube. Thus it is possible to solve the puzzle in configuration $t(C_0)$ in at most k moves. In other words, $(t(C_0), k)$ is a "yes" instance to the non-Group variant of the puzzle problem.

At this point it is also worth mentioning that the Rubik's Square with SQTM move model is a strictly more difficult puzzle than the Rubik's Square with STM move model:

Lemma 4.7. *If (C, k) is a "yes" instance to the SQTM Rubik's Cube problem, then it is also a "yes" instance to the STM Rubik's Cube problem. Similarly, if (t, k) is a "yes" instance to the Group SQTM Rubik's Cube problem, then it is also a "yes" instance to the Group STM Rubik's Cube problem.*

To prove this lemma, note that every move in the SQTM move model is a legal move in the STM move model. Then if configuration C can be solved in k or fewer SQTM moves, it can certainly also be solved in k or fewer STM moves. Similarly, if t can be inverted using at most k transformations corresponding to SQTM moves, then it can also be inverted using at most k transformations corresponding to STM moves.

4.2.5 Membership in NP

Consider the graph whose vertices are transformations in RS_n (or RC_n) and whose edges (a, b) connect transformations a and b for which $a^{-1}b$ is the transformation corresponding to a single move (under the standard Rubik's Square move model or under the STM or SQTM move model). It was shown in [11] that the diameter of this graph is $\Theta(\frac{n^2}{\log n})$. This means that any achievable transformation of the puzzle (any transformation in RS_n or RC_n) can be reached using a polynomial $p(n)$ number of moves.

Using this fact, we can build an NP algorithm solving the (Group) STM/SQTM Rubik's Cube and the (Group) Rubik's Square problems. In these problems, we are given k and either a starting configuration or a transformation, and we are asked whether it is possible to solve the configuration/invert the transformation in at most k moves. The NP algorithm can nondeterministically make $\min(k, p(n))$ moves and simply check whether this move sequence inverts the given transformation or solves the given puzzle configuration.

If any branch accepts, then certainly the answer to the problem is "yes" (since that branch's chosen sequence of moves is a solving/inverting sequence of moves of length at most k). On the other hand, if there is a solving/inverting sequence of moves of length at most k, then there is also one that has length both at most k and at most $p(n)$. This is because $p(n)$ is an upper bound on the diameter of the graph described above. Thus, if the answer to the problem is "yes", then there exists a solving/inverting sequence of moves of length at most $\min(k, p(n))$, and so at least one branch accepts. As desired, the algorithm described is correct. Therefore, we have established membership in NP for the problems in question.

4.3 Hamiltonicity variants

To prove the problems introduced above hard, we need to introduce several variants of the Hamiltonian cycle and path problems.

It is shown in [13] that the following problem is NP-complete.

Problem 4.8. *A* square grid graph *is a finite induced subgraph of the infinite square lattice. The* Grid Graph Hamiltonian Cycle *problem asks whether a given square grid graph with no degree-*1 *vertices has a Hamiltonian cycle.*

Starting with this problem, we prove that the following promise version of the grid graph Hamiltonian path problem is also NP-hard.

Problem 4.9. *The* Promise Grid Graph Hamiltonian Path *problem takes as input a square grid graph G and two specified vertices s and t with the promise that any Hamiltonian path*

in G has s and t as its start and end respectively. The problem asks whether there exists a Hamiltonian path in G.

The above problem is more useful, but it is still inconvenient in some ways. In particular, there is no conceptually simple way to connect a grid graph to a Rubik's Square or Rubik's Cube puzzle. It is the case, however, that every grid graph is actually a type of graph called a "cubical graph". Cubical graphs, unlike grid graphs, can be conceptually related to Rubik's Cubes and Rubik's Squares with little trouble.

So what is a cubical graph? Let H_m be the m dimensional hypercube graph; in particular, the vertices of H_m are the bitstrings of length m and the edges connect pairs of bitstrings whose Hamming distance is exactly one. Then a *cubical graph* is any induced subgraph of any hypercube graph H_m.

Notably, when embedding a grid graph into a hypercube, it is always possible to assign the bitstring label $00\ldots0$ to any vertex. Suppose we start with Promise Grid Graph Hamiltonian Path problem instance (G, s, t); then by embedding G into a hypercube graph, we can reinterpret this instance as an instance of the promise version of cubical Hamiltonian path:

Problem 4.10. *The* Promise Cubical Hamiltonian Path *problem takes as input a cubical graph whose vertices are length-m bitstrings $l_1, l_2, \ldots, l_n$ with the promise that (1) $l_n = 00\ldots0$ and (2) any Hamiltonian path in the graph has l_1 and l_n as its start and end respectively. The problem asks whether there exists a Hamiltonian path in the cubical graph. In other words, the problem asks whether it is possible to rearrange bitstrings $l_1, \ldots, l_n$ into a new order such that each bitstring has Hamming distance one from the next.*

In the following subsections, we will show the reductions used to prove Problems 4.9 and 4.10 hard.

4.3.1 Promise Grid Graph Hamiltonian Path is NP-hard

First, we reduce from the Grid Graph Hamiltonian Cycle problem to the Promise Grid Graph Hamiltonian Path problem.

Lemma 4.11. *The Promise Grid Graph Hamiltonian Path problem (Problem 4.9) is NP-hard.*

Proof. Consider an instance G of the Grid Graph Hamiltonian Cycle problem. Consider the vertices in the top row of G and let the leftmost vertex in this row be u. u has no neighbors on its left or above it, so it must have a neighbor to its right (since G has no degree-1 vertices). Let that vertex be u'. We can add vertices to G above u and u' as shown in figure 4-3 to obtain new grid graph G' in polynomial time. Note that two of the added vertices are labeled v and v'. Also note that the only edges that are added are those shown in the figure since no vertices in G are above u.

Notice that (G', v, v') is a valid instance of the Promise Grid Graph Hamiltonian Path problem. In particular, (G', v, v') satisfies the promise—any Hamiltonian path in G' must have v and v' as endpoints—since both v and v' have degree-1.

Below we show that (G', v, v') is a "yes" instance to the Promise Grid Graph Hamiltonian Path problem (i.e., G' has a Hamiltonian path) if and only if G is a "yes" instance to the Grid Graph Hamiltonian Cycle problem (i.e., G has a Hamiltonian cycle).

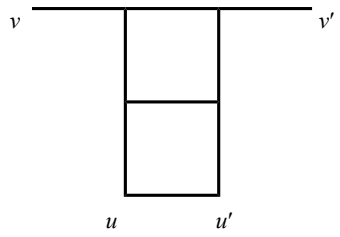


Figure 4-3: The vertices added to G to obtain G'.

First suppose G contains a Hamiltonian cycle. This cycle necessarily contains edge (u, u') because u has only two neighbors; removing this edge yields a Hamiltonian path from u' to u in G. This path can be extended by adding paths from v' to u' and from u to v into a Hamiltonian path in G' from v' to v.

On the other hand, suppose G' has a Hamiltonian path. Such a path must have v and v' as the two endpoints, and it is easy to show that the two short paths between u and v and between u' and v' must be the start and end of this path. In other words, if G' has a Hamiltonian path, then the central part of this path is a Hamiltonian path in G' between u and u'. Adding edge (u, u'), we obtain a Hamiltonian cycle in G.

By the above reduction, the Promise Grid Graph Hamiltonian Path problem is NP-hard. □

4.3.2 Promise Cubical Hamiltonian Path is NP-hard

Second, we reduce from the Promise Grid Graph Hamiltonian Path problem to the Promise Cubical Hamiltonian Path problem.

Theorem 4.12. *The Promise Cubical Hamiltonian Path problem (Problem 4.10) is NP-hard.*

Proof. Consider an instance (G, s, t) of the Promise Grid Graph Hamiltonian Path problem. Suppose G has m_r rows and m_c columns and n vertices.

Assign a bitstring label to each row and a bitstring label to each column. In particular, let the row labels from left to right be the following length $m_r - 1$ bitstrings: $000\ldots0$, $100\ldots0$, $110\ldots0$, …, and $111\ldots1$. Similarly, let the column labels from top to bottom be the following length $m_c - 1$ bitstrings: $000\ldots0$, $100\ldots0$, $110\ldots0$, …, and $111\ldots1$. Then assign each vertex a bitstring label of length $m = m_r + m_c - 2$ consisting of the concatenation of its row label followed by its column label.

Consider any two vertices. Their labels have Hamming distance one if and only if the vertices' column labels are the same and their row labels have Hamming distance one, or visa versa. By construction, two row/column labels are the same if and only if the two rows/columns are the same and they have Hamming distance one if and only if the two rows/columns are adjacent. Thus two vertices' labels have Hamming distance one if and only if the two vertices are adjacent in G.

In other words, we have expressed G as a cubical graph by assigning these bitstring labels to the vertices of G. In particular, suppose the vertices of G are $v_1, v_2, \ldots, v_n$ with $v_1 = s$ and $v_n = t$. Let l'_i be the label of v_i. Then the bitstrings $l'_1, l'_2, \ldots, l'_n$ specify the cubical graph that is G.

Define $l_i = l'_i \oplus l'_n$. Under this definition, the Hamming distance between l_i and l_j is the same as the Hamming distance between l'_i and l'_j. Therefore l_i has Hamming distance one

from l_j if and only if v_i and v_j are adjacent. Thus, the cubical graph specified by bitstrings $l_1, \ldots, l_n$ is also G. Note that the l_i bitstrings can be computed in polynomial time.

We claim that $l_1, \ldots, l_n$ is a valid instance of Promise Cubical Hamiltonian Path, i.e., this instance satisfies the promise of the problem. The first promise is that $l_n = 00\ldots0$; by definition, $l_n = l'_n \oplus l'_n = 00\ldots0$. The second promise is that any Hamiltonian path in the cubical graph specified by $l_1, l_2, \ldots, l_n$ has l_1 and l_n as its start and end. The cubical graph specified by $l_1, l_2, \ldots, l_n$ is the graph G with vertex l_i in the cubical graph corresponding to vertex v_i in G. In other words, the promise requested is that any Hamiltonian path in G must start and end in vertices $v_1 = s$ and $v_n = t$. This is guaranteed by the promise of the Promise Grid Graph Hamiltonian Path problem.

Since G is the graph specified by $l_1, l_2, \ldots, l_n$, the answer to the Promise Cubical Hamiltonian Path instance $l_1, l_2, \ldots, l_n$ is the same as the answer to the Promise Grid Graph Hamiltonian Path instance (G, s, t). Thus, the procedure converting (G, s, t) into $l_1, l_2, \ldots, l_n$, which runs in polynomial time, is a reduction proving that Promise Cubical Hamiltonian Path is NP-hard. □

4.4 (Group) Rubik's Square is NP-complete

4.4.1 Reductions

To prove that the Rubik's Square and Group Rubik's Square problems are NP-complete, we reduce from the Promise Cubical Hamiltonian Path problem of Section 4.3.2.

Suppose we are given an instance of the Promise Cubical Hamiltonian Path problem consisting of n bitstrings $l_1, \ldots, l_n$ of length m (with $l_n = 00\ldots0$). To construct a Group Rubik's Square instance we need to compute the value k indicating the allowed number of moves and construct the transformation $t \in RS_s$.

The value k can be computed directly as $k = 2n - 1$.

The transformation t will be an element of group RS_s where $s = 2(\max(m, n) + 2n)$. Define a_i for $1 \leq i \leq n$ to be $(x_1)^{(l_i)_1} \circ (x_2)^{(l_i)_2} \circ \cdots \circ (x_m)^{(l_i)_m}$ where $(l_i)_1, (l_i)_2, \ldots, (l_i)_m$ are the bits of l_i. Also define $b_i = (a_i)^{-1} \circ y_i \circ a_i$ for $1 \leq i \leq n$. Then we define t to be $a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$.

Outputting (t, k) completes the reduction from the Promise Cubical Hamiltonian Path problem to the Group Rubik's Square problem. To reduce from the Promise Cubical Hamiltonian Path problem to the Rubik's Square problem we simply output $(C_t, k) = (t(C_0), k)$. These reductions clearly run in polynomial time.

4.4.2 Intuition

The key idea that makes this reduction work is that the transformations b_i for $i \in \{1, \ldots, n\}$ all commute. This allows us to rewrite $t = a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$ with the b_is in a different order. If the order we choose happens to correspond to a Hamiltonian path in the cubical graph specified by $l_1, \ldots, l_n$, then when we explicitly write the b_is and a_1 in terms of x_js and y_is, most of the terms cancel. In particular, the number of remaining terms will be exactly k. Since we can write t as a combination of exactly k x_js and y_is, we can invert t using at most k x_js and y_is. In other words, if there is a Hamiltonian path in the cubical graph specified by $l_1, \ldots, l_n$, then (t, k) is a "yes" instance to the Group Rubik's Square problem.

In order to more precisely describe the cancelation of terms in t, we can consider just one local part: $b_i \circ b_{i'}$. We can rewrite this as $(a_i)^{-1} \circ y_i \circ a_i \circ (a_{i'})^{-1} \circ y_{i'} \circ a_{i'}$. The interesting

part is that $a_i \circ (a_{i'})^{-1}$ will cancel to become just one x_j. Note that

$$a_i \circ (a_{i'})^{-1} = (x_1)^{(l_i)_1} \circ (x_2)^{(l_i)_2} \circ \cdots \circ (x_m)^{(l_i)_m} \circ (x_1)^{-(l_{i'})_1} \circ (x_2)^{-(l_{i'})_2} \circ \cdots \circ (x_m)^{-(l_{i'})_m},$$

which we can rearrange as

$$(x_1)^{(l_i)_1-(l_{i'})_1} \circ (x_2)^{(l_i)_2-(l_{i'})_2} \circ \cdots \circ (x_m)^{(l_i)_m-(l_{i'})_m}.$$

Next, if b_i and $b_{i'}$ correspond to adjacent vertices l_i and $l_{i'}$, then $(l_i)_j - (l_{i'})_j$ is zero for all j except one for which $(l_i)_j - (l_{i'})_j = \pm 1$. Thus the above can be rewritten as $(x_j)^1$ or $(x_j)^{-1}$ for some specific j. Since $x_j = (x_j)^{-1}$ this shows that $(a_{i_1})^{-1} \circ a_{i_2}$ simplifies to x_j for some j.

This intuition is formalized in a proof in the following subsection.

4.4.3 Promise Cubical Hamiltonian Path solution → (Group) Rubik's Square solution

Lemma 4.13. *The transformations b_i all commute.*

Proof. Consider any such transformation b_i. The transformation b_i can be rewritten as $(a_i)^{-1} \circ y_i \circ a_i$. For any cubie not moved by the y_i middle term, the effect of this transformation is the same as the effect of transformation $(a_i)^{-1} \circ a_i = 1$. In other words, b_i only affects cubies that are moved by the y_i term. But y_i only affects cubies with y coordinate i. In general in a Rubik's Square, cubies with y coordinate i at some particular time will have y coordinate $\pm i$ at all times. Thus, all the cubies affected by b_i start in rows $\pm i$.

This is enough to see that the cubies affected by b_i are disjoint from those affected by b_j (for $j \neq i$). In other words, the transformations b_i all commute. □

Theorem 4.14. *If $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, then (t, k) is a "yes" instance to the Group Rubik's Square problem.*

Proof. Suppose $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem. Let m be the length of l_i and note that $l_n = 00\ldots0$ by the promise of the Promise Cubical Hamiltonian Path problem. Furthermore, since $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, there exists an ordering of these bitstrings $l_{i_1}, l_{i_2}, \ldots, l_{i_n}$ such that each consecutive pair of bitstrings is at Hamming distance one, $i_1 = 1$, and $i_n = n$ (with the final two conditions coming from the promise).

By Lemma 4.13, we know that $t = a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$ can be rewritten as

$$t = a_1 \circ b_{i_1} \circ b_{i_2} \circ \cdots \circ b_{i_n}.$$

Using the definition of b_i, we can further rewrite this as

$$t = a_1 \circ ((a_{i_1})^{-1} \circ y_{i_1} \circ a_{i_1}) \circ ((a_{i_2})^{-1} \circ y_{i_2} \circ a_{i_2}) \circ \cdots \circ ((a_{i_n})^{-1} \circ y_{i_n} \circ a_{i_n}),$$

or as

$$t = (a_1 \circ (a_{i_1})^{-1}) \circ y_{i_1} \circ (a_{i_1} \circ (a_{i_2})^{-1}) \circ y_{i_2} \circ (a_{i_2} \circ (a_{i_3})^{-1}) \circ \cdots \circ (a_{i_{n-1}} \circ (a_{i_n})^{-1}) \circ y_{i_n} \circ (a_{i_n}).$$

We know that $i_1 = 1$, and therefore that $a_1 \circ (a_{i_1})^{-1} = a_1 \circ (a_1)^{-1} = 1$ is the identity element. Similarly, we know that $i_n = n$ and therefore that $a_{i_n} = a_n = (x_1)^{(l_n)_1} \circ (x_2)^{(l_n)_2} \circ \cdots \circ (x_m)^{(l_n)_m} = (x_1)^0 \circ (x_2)^0 \circ \cdots \circ (x_m)^0 = 1$ is also the identity.

Thus we see that

$$t = y_{i_1} \circ (a_{i_1} \circ (a_{i_2})^{-1}) \circ y_{i_2} \circ (a_{i_2} \circ (a_{i_3})^{-1}) \circ \cdots \circ (a_{i_{n-1}} \circ (a_{i_n})^{-1}) \circ y_{i_n}.$$

Consider the transformation $a_{i_p} \circ (a_{i_{p+1}})^{-1}$. This transformation can be written as

$$a_{i_p} \circ (a_{i_{p+1}})^{-1} = (x_1)^{(l_{i_p})_1} \circ (x_2)^{(l_{i_p})_2} \circ \cdots \circ (x_m)^{(l_{i_p})_m} \circ (x_1)^{-(l_{i_{p+1}})_1} \circ (x_2)^{-(l_{i_{p+1}})_2} \circ \cdots \circ (x_m)^{-(l_{i_{p+1}})_m}.$$

Because x_u always commutes with x_v, we can rewrite this as

$$a_{i_p} \circ (a_{i_{p+1}})^{-1} = (x_1)^{(l_{i_p})_1 - (l_{i_{p+1}})_1} \circ (x_2)^{(l_{i_p})_2 - (l_{i_{p+1}})_2} \circ \cdots \circ (x_m)^{(l_{i_p})_m - (l_{i_{p+1}})_m}.$$

Since l_{i_p} differs from $l_{i_{p+1}}$ in only one position, call it j_p, we see that $(l_{i_p})_j - (l_{i_{p+1}})_j$ is zero unless $j = j_p$, and is ± 1 in that final case. This is sufficient to show that $a_{i_p} \circ (a_{i_{p+1}})^{-1} = (x_{j_p})^{\pm 1} = x_{j_p}$.

Thus we see that

$$t = y_{i_1} \circ x_{j_1} \circ y_{i_2} \circ x_{j_2} \circ \cdots \circ x_{j_{n-1}} \circ y_{i_n},$$

or (by left multiplying) that

$$1 = y_{i_n}^{-1} \circ x_{j_{n-1}}^{-1} \circ \cdots \circ x_{j_2}^{-1} \circ y_{i_2}^{-1} \circ x_{j_1}^{-1} \circ y_{i_1}^{-1} \circ t = y_{i_n} \circ x_{j_{n-1}} \circ \cdots \circ x_{j_2} \circ y_{i_2} \circ x_{j_1} \circ y_{i_1} \circ t.$$

We see that t can be reversed by $k = 2n - 1$ moves of the form x_j or y_i, or in other words that (t, k) is a "yes" instance to the Group Rubik's Square problem. □

Corollary 4.15. *If $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, then (C_t, k) is a "yes" instance to the Rubik's Square problem.*

Proof. This follows immediately from Theorem 4.14 and Lemma 4.5. □

4.4.4 Coloring of C_t

In order to show the other direction of the proof, it will be helpful to consider the coloring of the stickers on the top and bottom faces of the Rubik's Square. In particular, if we define $b = b_1 \circ \cdots \circ b_n$ (so that $t = a_1 \circ b$), then it will be very helpful for us to know the colors of the top and bottom stickers in configuration $C_b = b(C_0)$.

Consider for example the instance of Promise Cubical Hamiltonian Path with $n = 5$ and $m = 3$ defined below:

$$l_1 = 011$$
$$l_2 = 110$$
$$l_3 = 111$$
$$l_4 = 100$$
$$l_5 = 000$$

For this example, C_0 is an $s \times s$ Rubik's Square with $s = 2(\max(m, n) + 2n) = 24$.

To describe configuration C_b, we need to know the effect of transformation b_i. For example, Figure 4-4 shows the top face of a Rubik's Square in configurations C_0, $a_2(C_0)$, $(y_2 \circ a_2)(C_0)$, and $b_2(C_0) = ((a_2)^{-1} \circ y_2 \circ a_2)(C_0)$ where a_2 and y_2 are defined in terms of $l_2 = 110$ as in the reduction.

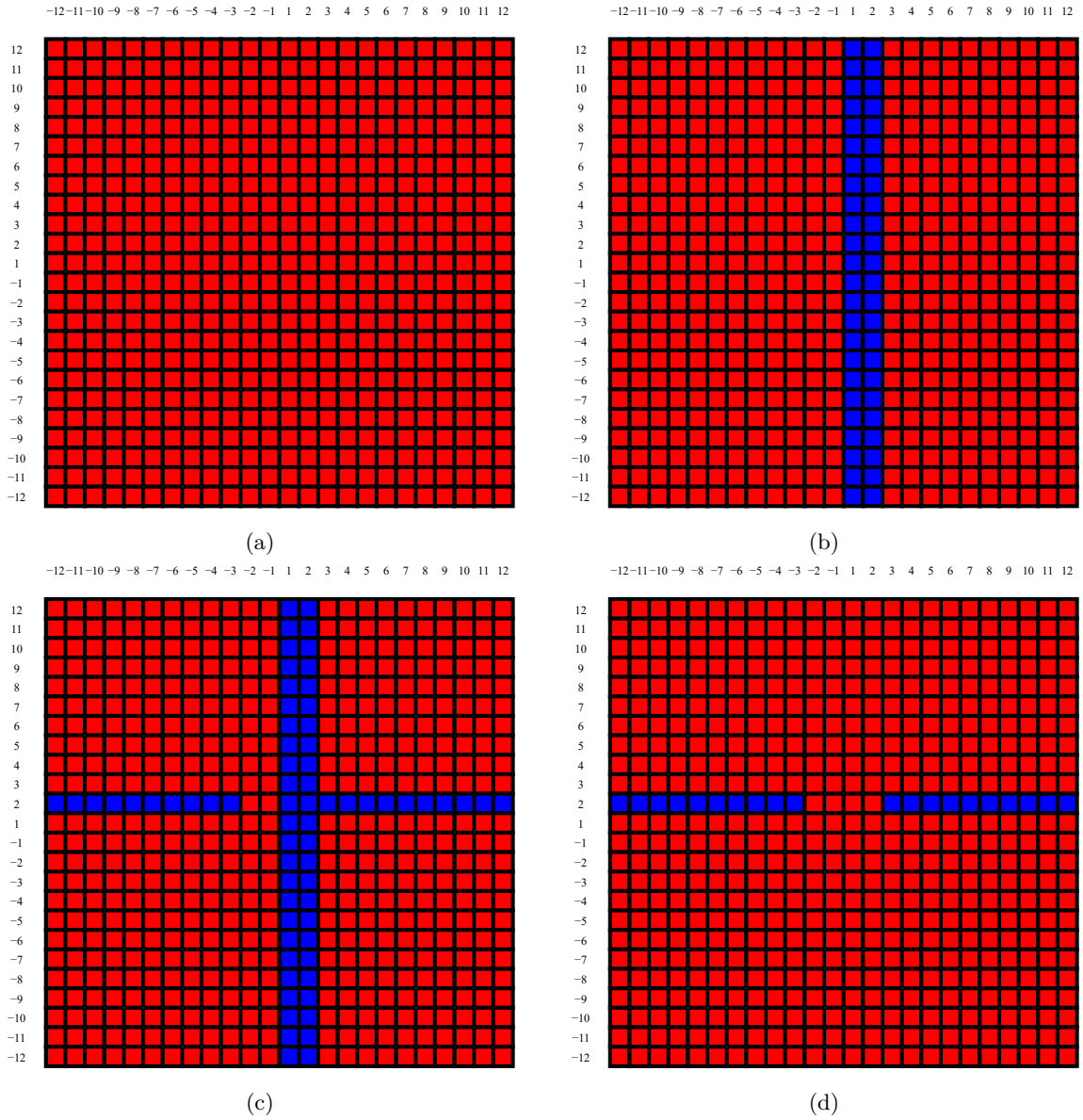


Figure 4-4: Applying b_2 to C_0 step by step (only top face shown).

The exact behavior of a Rubik's Square due to b_i is described by the following lemma:

Lemma 4.16. *Suppose $i \in \{1, \ldots, n\}$, and $c, r \in \{1, \ldots, s/2\}$. Then*

1. *if $r = i$ and $c \leq m$ such that bit c of l_i is 1, then b_i swaps the cubies in positions $(c, -r)$ and $(-c, r)$ without flipping either;*
2. *if $r = i$ and either $c > m$ or $c \leq m$ and bit c of l_i is 0, then b_i swaps the cubies in positions (c, r) and $(-c, r)$ and flips them both;*
3. *all other cubies are not moved by b_i.*

Proof. As noted in the proof of Lemma 4.13, a cubie is affected by $b_i = (a_i)^{-1} \circ y_i \circ a_i$ if and only if it is moved by the y_i term.

Note also that $(a_i)^{-1} = a_i$ only moves cubies within their columns and only for columns c for which bit c of l_i is 1. One consequence is that a cubie can only be moved by a_i if its column index is positive. Any cubie moved by the y_i term will have a column index of different signs before and after the y_i move, so as a consequence such a cubie cannot be moved by both a_i and $(a_i)^{-1}$.

Thus there are three possibilities for cubies that are moved by b_i: (1) the cubie is moved only by y_i, (2) the cubie is moved by a_i and then by y_i, and (3) the cubie is moved by y_i and then by $(a_i)^{-1}$.

Consider any cubie of type (1) whose coordinates have absolute values c and r. Since the cubie is moved by y_i, we know that $r = i$. Since it is not moved by either a_i or $(a_i)^{-1}$, we know that the cubie's column index both before and after the move is not one of the column indices affected by a_i. But these two column indices are c and $-c$ (in some order). Therefore it must not be the case that bit c of l_i is 1. Also note that cubies of this type are flipped exactly once. Putting that together, we see that if $c \in \{1, \ldots, s/2\}$, $r = i$, and it is not the case that bit c of l_i exists and is 1, then b_i swaps the cubies in positions (c, r) and $(-c, r)$ and flips them both.

Consider any cubie of type (2) whose coordinates have absolute values c and r. Since the cubie is first moved by a_i and then by y_i, we know that $r = i$ and that $c \leq m$ with bit c of l_i equal to 1. Furthermore, the cubie must have started in position $(c, -r)$, then moved to position (c, r) by a_i, and then moved to position $(-c, r)$ by y_i. Since this cubie is flipped twice, it is overall not flipped.

Consider on the other hand any cubie of type (3) whose coordinates have absolute values c and r. Since the cubie is first moved by y_i and then by $(a_i)^{-1} = a_i$, we know that $r = i$ and that $c \leq m$ with bit c of l_i equal to 1. Furthermore, the cubie must have started in position $(-c, r)$, then moved to position (c, r) by y_i, and then moved to position $(c, -r)$ by a_i. Since this cubie is flipped twice, it is overall not flipped.

Putting that together, we see that if $r = i$, and bit c of l_i is 1, then b_i swaps the cubies in positions $(c, -r)$ and $(-c, r)$ without flipping either.

This covers the three types of cubies that are moved by b_i. All other cubies remain in place. □

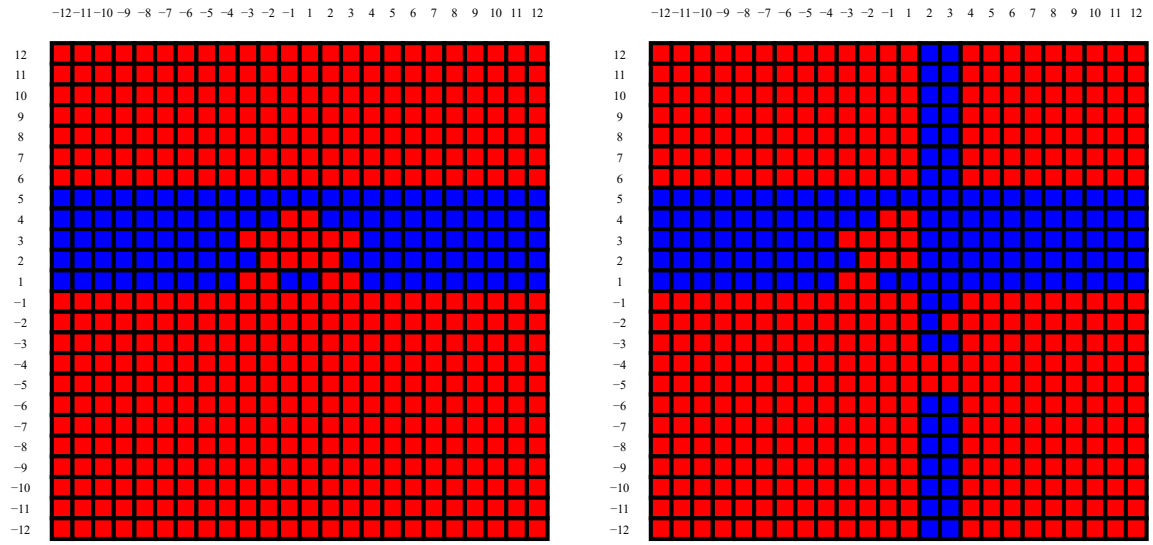
We can apply the above to figure out the effect of transformation $b_1 \circ b_2 \circ \cdots \circ b_n$ on configuration C_0. In particular, that allows us to learn the coloring of configuration C_b.

Theorem 4.17. *In C_b, a cubie has top face blue if and only if it is in position (c, r) such that $1 \leq r \leq n$ and either $|c| > m$ or $|c| \leq m$ and bit $|c|$ of l_r is 0.*

Proof. C_b is obtained from C_0 by applying transformation $b_1 \circ b_2 \circ \cdots \circ b_n$. A cubie has top face blue in C_b if and only if transformation $b_1 \circ b_2 \circ \cdots \circ b_n$ flips that cubie an odd number of times. Each b_i affects a disjoint set of cubies. Thus, among the cubies affected by some particular b_i, the only ones that end up blue face up are the ones that are flipped by b_i. By Lemma 4.16, these are the cubies in row i with column c such that it is not the case that bit $|c|$ of l_i is 1. Tallying up those cubies over all the b_is yields exactly the set of blue-face-up cubies given in the theorem statement. □

This concludes the description of C_b in terms of colors. The coloring of configuration C_t—the configuration that is actually obtained by applying the reduction to $l_1, \ldots, l_n$—can be obtained from the coloring of configuration C_b by applying transformation a_1.

Applying Theorem 4.17 to the previously given example, we obtain the coloring of the Rubik's Square in configuration C_b as shown in Figure 4-5a. Note that the $n \times m$ grid of bits comprising $l_1, \ldots, l_n$ is actually directly encoded in the coloring of a section of the Rubik's Square. In addition, the coloring of the Rubik's Square in configuration C_t is shown for the same example in Figure 4-5b.



(a) The top face of C_b for the example input $l_1, \ldots, l_n$.

(b) The top face of C_t for the example input $l_1, \ldots, l_n$.

Figure 4-5: The coloring of the Rubik's Square for the example input $l_1, \ldots, l_n$.

4.4.5 (Group) Rubik's Square solution → Promise Cubical Hamiltonian Path solution

Below, we prove the following theorem:

Theorem 4.18. *If (C_t, k) is a "yes" instance to the Rubik's Square problem, then $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

By Lemma 4.5, this will immediately also imply the following corollary:

Corollary 4.19. *If (t, k) is a "yes" instance to the Group Rubik's Square problem, then $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

To prove the theorem, we consider a hypothetical solution to the (C_t, k) instance of the Rubik's Square problem. A solution consists of a sequence of Rubik's Square moves $m_1, \ldots, m_{k'}$ with $k' \leq k$ such that $C' = (m_{k'} \circ \cdots \circ m_1)(C_t)$ is a solved configuration of the Rubik's Square. Throughout the proof, we will use only the fact that move sequence $m_1, \ldots, m_{k'}$ solves the top and bottom faces of the Rubik's Square in configuration C_t.

The main idea of the proof relies on three major steps. In the first step, we show that $m_1, \ldots, m_{k'}$ must flip row i an odd number of times if $i \in \{1, \ldots, n\}$, and an even number of times otherwise.

We then define set $O \subseteq \{1, \ldots, n\}$ (where O stands for "one") to be the set of indices i such that there is exactly one index-i row move. Clearly, in order to satisfy the parity constraints, every $i \in O$ must have one row i move and zero row $-i$ moves in $m_1, \ldots, m_{k'}$.

The second step of the proof is to show that, if $i_1, i_2 \in O$, then the number of column moves in $m_1, \ldots, m_{k'}$ between the single flip of row i_1 and the single flip of row i_2 is at least the Hamming distance between l_{i_1} and l_{i_2}.

The final step of the proof is a counting argument. There are four types of moves in $m_1, \ldots, m_{k'}$:

1. index-i row moves with $i \in O$ (all of which are positive moves as shown above),
2. index-i row moves with $i \in \{1, \ldots, n\} \setminus O$,
3. column moves, and
4. index-i row moves with $i \notin \{1, \ldots, n\}$.

For each $i \in O$, there is exactly one index-i move by definition of O. Therefore the number of type-1 moves is exactly $|O|$.

For each i in $\{1, \ldots, n\} \setminus O$, the number of index-$i$ row moves is odd by the parity constraint. Furthermore, by the definition of O, this number is not one. Thus each i in $\{1, \ldots, n\} \setminus O$ contributes at least three moves. Therefore the number of type-2 moves is at least $3(|\{1, \ldots, n\} \setminus O|) = 3(n - |O|)$.

Consider the moves of rows i with $i \in O$. Since the l_is are all distinct, there must be at least one column move between every consecutive pair of such moves. Thus the total number of type-3 moves (column moves) is at least $|O| - 1$. Furthermore, the number of type-3 moves is $|O| - 1$ if and only if the consecutive pairs of row $i \in O$ moves have exactly one column move between them. Such a pair of is has exactly one column move between the two row-i moves only if the corresponding pair of l_is is at Hamming distance one. Therefore, if we consider the l_is for $i \in O$ in the order in which row-i moves occur in $m_1, \ldots, m_{k'}$, then the number of type-3 moves is exactly $|O| - 1$ if and only if those l_is in that order have each l_i at Hamming distance exactly one from the next (and more otherwise).

The number of type-4 moves is at least 0.

Adding these bounds up, we see that there are at least $(|O|) + 3(n - |O|) + (|O| - 1) + 0 = 3n - 1 - |O| = k + (n - |O|)$ moves. Since $n - |O| \geq 0$ and the number of moves is at most k, we can conclude that (1) $|O| = n$ and (2) the number of moves of each type is exactly the minimum possible computed above. Since $|O| = n$ we know that $O = \{1, \ldots, n\}$. But then looking at the condition for obtaining the minimum possible number of type-3 moves, we see that the l_is for $i \in O = \{1, \ldots, n\}$ in the order in which row-i flips occur in $m_1, \ldots, m_{k'}$ are each at Hamming distance exactly one from the next. Thus, there is a reordering of $l_1, \ldots, l_n$ in which each l_i is Hamming distance one from the next; in other words, the cubical graph specified by bitstrings $l_1, \ldots, l_n$ has a Hamiltonian path and $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.

All that's left is to complete the first two steps of the proof. We prove these two steps in the lemmas below:

Lemma 4.20. *Move sequence $m_1, \ldots, m_{k'}$ must flip row i an odd number of times if $i \in \{1, \ldots, n\}$, and an even number of times otherwise.*

Proof. Consider the transformation

$$m_{k'} \circ \cdots \circ m_1 \circ t = m_{k'} \circ \cdots \circ m_1 \circ a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n.$$

This transformation, while not necessarily the identity transformation, must transform C_0 into another solved Rubik's Square configuration C'.

Consider the $2n = k + 1$ indices $\max(m, n) + 1, \ldots, \max(m, n) + 2n$. At least one such index i must exist for which no move in $m_1, \ldots, m_{k'}$ is an index-i move. Let u be such an index.

Consider the effect of transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$ on the cubie in position (u, u). If we write $t = a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$ as a sequence of x_js and $y_{i'}$s (using the definitions of a_1 and b_i), then every move in t flips rows and columns with indices of absolute value at most $\max(m, n)$. Thus no term in the transformation $(m_{k'} \circ \cdots \circ m_1 \circ a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n)$ flips row or column u. We conclude that the cubie in position (u, u) is unmoved by this transformation. Applying this transformation to C_0 yields C'. So since this cubie starts with top sticker red in configuration C_0, the final configuration C' also has this cubie's top sticker red. Since C' is a solved configuration, the entire top face in C' must be red.

Next consider the cubie in position (u, r) for any r. Since no row or column with index $\pm u$ is ever flipped in transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$, this cubie is only ever affected by flips of row r. Furthermore, every flip of row r flips this cubie and therefore switches the color of its top face. Since the transformation in question converts configuration C_0 into configuration C', both of which have every cubie's top face red, the row in question must be flipped an even number of times.

For $i \in \{1, \ldots, n\}$, the tranformation $a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$, when written out fully in terms of $y_{i'}$s and x_js, includes exactly one flip of row y_i. Thus move sequence $m_1, \ldots, m_{k'}$ must flip each of these rows an odd number of times. Similarly, for $i \notin \{1, \ldots, n\}$, the tranformation $a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$, when written out fully in terms of $y_{i'}$s and x_js, does not include any flips of row y_i at all. Thus move sequence $m_1, \ldots, m_{k'}$ must flip each of these rows an even number of times. □

Lemma 4.21. *If $i_1, i_2 \in O$ (with $i_1 \neq i_2$), then the number of column moves x_j between the unique y_{i_1} and y_{i_2} moves in sequence $m_1, \ldots, m_{k'}$ is at least the Hamming distance between l_{i_1} and l_{i_2}.*

Proof. We will prove the following useful fact below: if $i_1, i_2 \in O$ (with $i_1 \neq i_2$) and $j \in \{1, 2, \ldots, m\}$ such that the top colors of the cubies in locations (j, i_1) and (j, i_2) are different in configuration C_b, then there must be at least one index-j column move in between the unique y_{i_1} and y_{i_2} moves in sequence $m_1, \ldots, m_{k'}$.

We know from Theorem 4.17 that, if $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$, then the top color of the cubie in location (j, i) of configuration C_b is red if and only if $(l_i)_j = 1$. Thus, if l_{i_1} and l_{i_2} differ in bit j, then in configuration C_b one of the two cubies in positions (j, i_1) and (j, i_2) will have top face red and the other will have top face blue. Applying the above useful fact, we see that at least one index-j column move will occur in sequence $m_1, \ldots, m_{k'}$ between the unique y_{i_1} and y_{i_2} moves. Since this column move has index $\pm j$, every difference in l_{i_1} and l_{i_2} will contribute at least one distinct column move between the unique y_{i_1} and y_{i_2} moves. Assuming the useful fact, we can conclude that the number of column moves between the unique y_{i_1} and y_{i_2} moves is at least the Hamming distance between l_{i_1} and l_{i_2}, as desired.

We now prove the useful fact by contradiction. Assume that the useful fact is false, i.e., that there exists some $i_1, i_2 \in O$ and $j \in \{1, 2, \ldots, m\}$ such that the top colors of the cubies in locations (j, i_1) and (j, i_2) are different in C_b and such that no index-j column move is made between the unique y_{i_1} and y_{i_2} moves in sequence $m_1, \ldots, m_{k'}$.

Consider these two cubies. Starting in configuration C_b, we can reach configuration C' by applying transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1 = m_{k'} \circ \cdots \circ m_1 \circ (x_1)^{(l_1)_1} \circ (x_2)^{(l_1)_2} \circ \cdots \circ (x_3)^{(l_1)_m}$. Note that this transformation consists of some (but not necessarily all) of the moves $x_1, x_2, \ldots, x_m$ followed by the move sequence $m_1, \ldots, m_{k'}$. We will consider the effect of this transformation on the two cubies.

Since the two cubies start in locations (j, i_1) and (j, i_2), the only moves that could ever affect these cubies are of the forms x_j, x_{-j}, y_{i_1}, y_{-i_1}, y_{i_2}, and y_{-i_2}. Furthermore, by the definition of O, no moves of the form y_{-i_1} or y_{-i_2} occur and the moves y_{i_1} and y_{i_2} each occur exactly once. Finally, we have by assumption that no moves of the form x_j or x_{-j} (index-j column moves) occur between moves y_{i_1} and y_{i_2}.

Putting these facts together, we see that the effect of transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1$ on these two cubies is exactly the same as the effect of some transformation of the following type: (1) some number of moves of the form x_j or x_{-j}, followed by (2) the two moves y_{i_1} and y_{i_2} in some order, followed by (3) some number of moves of the form x_j or x_{-j}.

Consider the effect of any such transformation on the two cubies. In step (1), each move of the form x_j or x_{-j} either flips both cubies (since they both start in column j) or flips neither, so the two cubies are each flipped an equal number of times. Furthermore, the row index of the two cubies is either positive for both or negative for both at all times throughout step (1). In step (2), either each of the two cubies is flipped exactly once (if their row indices at the start of step (2) are both positive) or neither of the two cubies is flipped at all (if their row indixes at the start of step (2) are negative); again, the number of flips is the same. Finally, in step (3), both cubies are in the same column (column j if they were not flipped in step (2) and column $-j$ if they were), so each move of the form x_j or x_{-j} either flips both cubies or flips neither; the two cubies are flipped an equal number of times. Thus we see that the two cubies are flipped an equal number of times by such a transformation.

We can conclude that the two cubies are flipped an equal number of times by transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1$. In configuration C_b, the two cubies have different colors on their top faces, so after transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1$ flips each of the two cubies an equal number of times, the resulting configuration still has different colors on the top faces of the two cubies. But the resulting configuration is C', which has red as the top face color of every cubie. Thus we have our desired contradiction. Therefore the useful fact is true and the desired result holds. □

4.4.6 Conclusion

Theorems 4.14 and 4.18 and Corollaries 4.15 and 4.19 show that the polynomial time reductions given are answer preserving. As a result, we conclude that

Theorem 4.22. *The Rubik's Square and Group Rubik's Square problems are NP-complete.*

4.5 (Group) STM/SQTM Rubik's Cube is NP-complete

4.5.1 Reductions

Below, we introduce the reductions used for the Rubik's Cube case. These reductions very closely mirror the Rubik's Square case, and the intuition remains exactly the same: the b_i terms commute, and so if the input Promise Cubical Hamiltonian Path instance is a "yes" instance then the b_is can be reordered so that all but k moves in the definition of t will cancel; therefore in that case t can be both enacted and reversed in k moves.

There are, however, several notable differences from the Rubik's Square case. The first difference is that in a Rubik's Cube, the moves x_i, y_i, and z_i are all quarter turn rotations rather than self-inverting row or column flips. One consequence is that unlike in the Rubik's Square case, the term a_i does not have the property that $(a_i)^{-1} = a_i$. A second difference is that in a Rubik's Square, the rows never become columns or visa versa. In a Rubik's Cube on the other hand, rotation of the faces can put rows of stickers that were once aligned parallel to one axis into alignment with another axis. To avoid allowing a solution of the puzzle due to this fact in the absence of a solution to the input Promise Cubical Hamiltonian Path instance, the slices in this construction which take the role of rows 1 through n in the Rubik's Square case and the slices which take the role of columns 1 through m in the Rubik's Square case will be assigned entirely distinct indices.

To prove that the STM/SQTM Rubik's Cube and Group STM/SQTM Rubik's Cube problems are NP-complete, we reduce from the Promise Cubical Hamiltonian Path problem of Section 4.3.2 as described below.

Suppose we are given an instance of the Promise Cubical Hamiltonian Path problem consisting of n biststrings $l_1, \ldots, l_n$ of length m (with $l_n = 00\ldots0$). To construct a Group STM/SQTM Rubik's Square instance we need to compute the value k indicating the allowed number of moves and construct the transformation t in RC_s.

The value k can be computed directly as $k = 2n - 1$.

The transformation t will be an element of group RC_s where $s = 6n + 2m$. Define a_i for $1 \leq i \leq n$ to be $(x_1)^{(l_i)_1} \circ (x_2)^{(l_i)_2} \circ \cdots \circ (x_m)^{(l_i)_m}$ where $(l_i)_1, (l_i)_2, \ldots, (l_i)_m$ are the bits of l_i. Also define $b_i = (a_i)^{-1} \circ z_{m+i} \circ a_i$ for $1 \leq i \leq n$. Then we define t to be $a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$.

Outputting (t, k) completes the reduction from the Promise Cubical Hamiltonian Path problem to the Group STM/SQTM Rubik's Cube problem. To reduce from the Promise Cubical Hamiltonian Path problem to the STM/SQTM Rubik's Cube problem we simply output $(C_t, k) = (t(C_0), k)$. As with the Rubik's Square case, these reductions are clearly polynomial time reductions.

4.5.2 Promise Cubical Hamiltonian Path solution → (Group) STM/SQTM Rubik's Cube solution

In this section, we prove one direction of the answer preserving property of the reductions. This proof is not substantively different from the proof of the first direction for the Rubik's Square problems (in Section 4.4.3). The differences in these proofs are all minor details that are only present to account for the differences (listed above) between the Rubik's Square and Rubik's Cube reductions.

Lemma 4.23. *The transformations b_i all commute.*

Proof. Consider any such transformation b_i. The transformation b_i can be rewritten as $(a_i)^{-1} \circ z_{m+i} \circ a_i$. For any cubie not moved by the z_{m+i} middle term, the effect of this transformation is the same as the effect of transformation $(a_i)^{-1} \circ a_i = 1$. In other words, b_i only affects cubies that are moved by the z_{m+i} term.

A cubie affected by this term was either moved into the z slice with index $(m + i)$ by a_i or was already there. a_i consists of some number of clockwise x turns. Thus, in order to be moved into a position with $z = (m + i)$, a cubie would have to start in a position with $y = -(m + i)$ on the $+z$ face or in a position with $y = (m + i)$ on the $-z$ face.

Thus, the cubies affected by b_i must either have y coordinate $\pm(m + i)$ and lie on one of the $\pm z$ faces or have z coordinate $(m + i)$ and lie on one of the other four faces. This is

enough to see that the cubies affected by b_i are disjoint from those affected by b_j (for $j \neq i$). In other words, the transformations b_i all commute. □

Theorem 4.24. *If $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, then (t, k) is a "yes" instance to the Group SQTM Rubik's Cube problem.*

Proof. Suppose $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem. Let m be the length of l_i and note that $l_n = 00\ldots0$ by the promise of the Promise Cubical Hamiltonian Path problem. Furthermore, since $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, there exists an ordering of these bitstrings $l_{i_1}, l_{i_2}, \ldots, l_{i_n}$ such that each consecutive pair of bitstrings is at Hamming distance one, $i_1 = 1$, and $i_n = n$ (with the final two conditions coming from the promise).

By Lemma 4.23, we know that $t = a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$ can be rewritten as

$$t = a_1 \circ b_{i_1} \circ b_{i_2} \circ \cdots \circ b_{i_n}.$$

Using the definition of b_i, we can further rewrite this as

$$t = a_1 \circ ((a_{i_1})^{-1} \circ z_{m+i_1} \circ a_{i_1}) \circ ((a_{i_2})^{-1} \circ z_{m+i_2} \circ a_{i_2}) \circ \cdots \circ ((a_{i_n})^{-1} \circ z_{m+i_n} \circ a_{i_n}),$$

or as

$$t = (a_1 \circ (a_{i_1})^{-1}) \circ z_{m+i_1} \circ (a_{i_1} \circ (a_{i_2})^{-1}) \circ z_{m+i_2} \circ (a_{i_2} \circ (a_{i_3})^{-1}) \circ \cdots \circ (a_{i_{n-1}} \circ (a_{i_n})^{-1}) \circ z_{m+i_n} \circ (a_{i_n}).$$

We know that $i_1 = 1$, and therefore that $a_1 \circ (a_{i_1})^{-1} = a_1 \circ (a_1)^{-1} = 1$ is the identity element. Similarly, we know that $i_n = n$ and therefore that $a_{i_n} = a_n = (x_1)^{(l_n)_1} \circ (x_2)^{(l_n)_2} \circ \cdots \circ (x_m)^{(l_n)_m} = (x_1)^0 \circ (x_2)^0 \circ \cdots \circ (x_m)^0 = 1$ is also the identity.

Thus we see that

$$t = z_{m+i_1} \circ (a_{i_1} \circ (a_{i_2})^{-1}) \circ z_{m+i_2} \circ (a_{i_2} \circ (a_{i_3})^{-1}) \circ \cdots \circ (a_{i_{n-1}} \circ (a_{i_n})^{-1}) \circ z_{m+i_n}.$$

Consider the transformation $a_{i_p} \circ (a_{i_{p+1}})^{-1}$. This transformation can be written as

$$a_{i_p} \circ (a_{i_{p+1}})^{-1} = (x_1)^{(l_{i_p})_1} \circ (x_2)^{(l_{i_p})_2} \circ \cdots \circ (x_m)^{(l_{i_p})_m} \circ (x_1)^{-(l_{i_{p+1}})_1} \circ (x_2)^{-(l_{i_{p+1}})_2} \circ \cdots \circ (x_m)^{-(l_{i_{p+1}})_m}.$$

Because x_u always commutes with x_v, we can rewrite this as

$$a_{i_p} \circ (a_{i_{p+1}})^{-1} = (x_1)^{(l_{i_p})_1 - (l_{i_{p+1}})_1} \circ (x_2)^{(l_{i_p})_2 - (l_{i_{p+1}})_2} \circ \cdots \circ (x_m)^{(l_{i_p})_m - (l_{i_{p+1}})_m}.$$

Since l_{i_p} differs from $l_{i_{p+1}}$ in only one position, call it j_p, we see that $(l_{i_p})_j - (l_{i_{p+1}})_j$ is zero unless $j = j_p$, and is ± 1 in that final case. This is sufficient to show that $a_{i_p} \circ (a_{i_{p+1}})^{-1} = (x_{j_p})^{s_p}$ where $s_p = \pm 1$.

Thus we see that

$$t = z_{m+i_1} \circ (x_{j_1})^{s_1} \circ z_{m+i_2} \circ (x_{j_2})^{s_2} \circ \cdots \circ (x_{j_{n-1}})^{s_{n-1}} \circ z_{m+i_n},$$

or (by left multiplying) that

$$(z_{m+i_n})^{-1} \circ (x_{j_{n-1}})^{-s_{n-1}} \circ \cdots \circ (x_{j_2})^{-s_2} \circ (z_{m+i_2})^{-1} \circ (x_{j_1})^{-s_1} \circ (z_{m+i_1})^{-1} \circ t = 1.$$

We see that t can be reversed by $k = 2n - 1$ terms of the form $(z_i)^{-1}$, x_j, and $(x_j)^{-1}$, which

are all SQTM moves. In other words, (t, k) is a "yes" instance to the Group SQTM Rubik's Cube problem. □

Corollary 4.25. *If $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, then (C_t, k) is a "yes" instance to the STM/SQTM Rubik's Cube problem and (t, k) is a "yes" instance to the Group STM/SQTM Rubik's Cube problem.*

Proof. This follows immediately from Theorem 4.24 and Lemmas 4.6 and 4.7. □

4.5.3 Coloring of C_t

As in the Rubik's Square case, it will be helpful for the second direction of the proof to know the coloring of the Cube's configuration. As before, we define $b = b_1 \circ \cdots \circ b_n$ (so that $t = a_1 \circ b$) and determine the colors of the stickers in configuration $C_b = b(C_0)$.

Consider the example instance of Promise Cubical Hamiltonian Path with $n = 5$ and $m = 3$ introduced in the Rubik's Square section and reproduced below:

$$l_1 = 011$$
$$l_2 = 110$$
$$l_3 = 111$$
$$l_4 = 100$$
$$l_5 = 000$$

For this example instance, the Rubik's Cube configuration produced by the reduction is an $s \times s \times s$ Rubik's Cube with $s = 2m + 6n = 36$. Furthermore, the coloring of the stickers in C_b for this example is shown in Figure 4-6. Note that the $n \times m$ grid of bits comprising $l_1, \ldots, l_n$ is actually directly encoded in the coloring of each face.

In this section, we prove the following useful theorem, which formalizes the pattern of colors from the example (Figure 4-6):

Theorem 4.26. *In C_b, the stickers have the following coloring:*

- *$+z$: The stickers on the $+z$ face with (x, y) coordinates $(j, -(m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all red. All other stickers are white.*
- *$-z$: The stickers on the $-z$ face with (x, y) coordinates $(j, -(m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all orange. All other stickers are blue.*
- *$+y$: The stickers on the $+y$ face with (x, z) coordinates $(j, (m + i))$ where $i \in \{1, \ldots, n\}$ and either l_i doesn't have a jth bit (i.e. $j < 0$ or $j > m$) or the jth bit of l_i is zero are all red. All other stickers are green.*
- *$-y$: The stickers on the $-y$ face with (x, z) coordinates $(j, (m + i))$ where $i \in \{1, \ldots, n\}$ and either l_i doesn't have a jth bit (i.e. $j < 0$ or $j > m$) or the jth bit of l_i is zero are all orange. All other stickers are yellow.*
- *$+x$: The stickers on the $+x$ face with (y, z) coordinates $(-j, (m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all white. All other stickers with z coordinate in $\{1, \ldots, n\}$ are green. All other stickers are orange.*

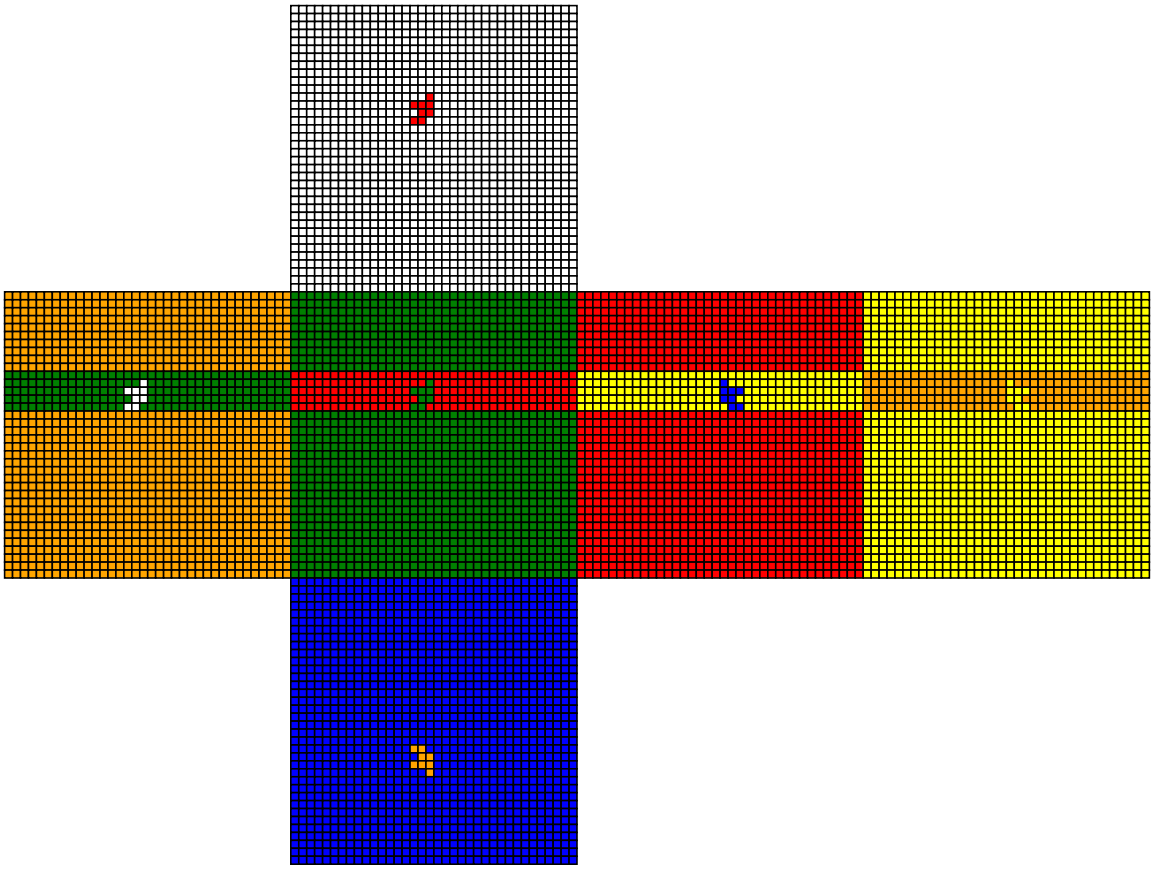


Figure 4-6: The faces of C_b for the example input $l_1, \ldots, l_n$. In this figure, the top and bottom faces are the $+z$ and $-z$ faces, while the faces in the vertical center of the figure are the $+x$, $+y$, $-x$, and $-y$ faces from left to right.

$-x$: *The stickers on the $-x$ face with (y, z) coordinates $(-j, (m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all blue. All other stickers with z coordinate in $\{1, \ldots, n\}$ are yellow. All other stickers are red.*

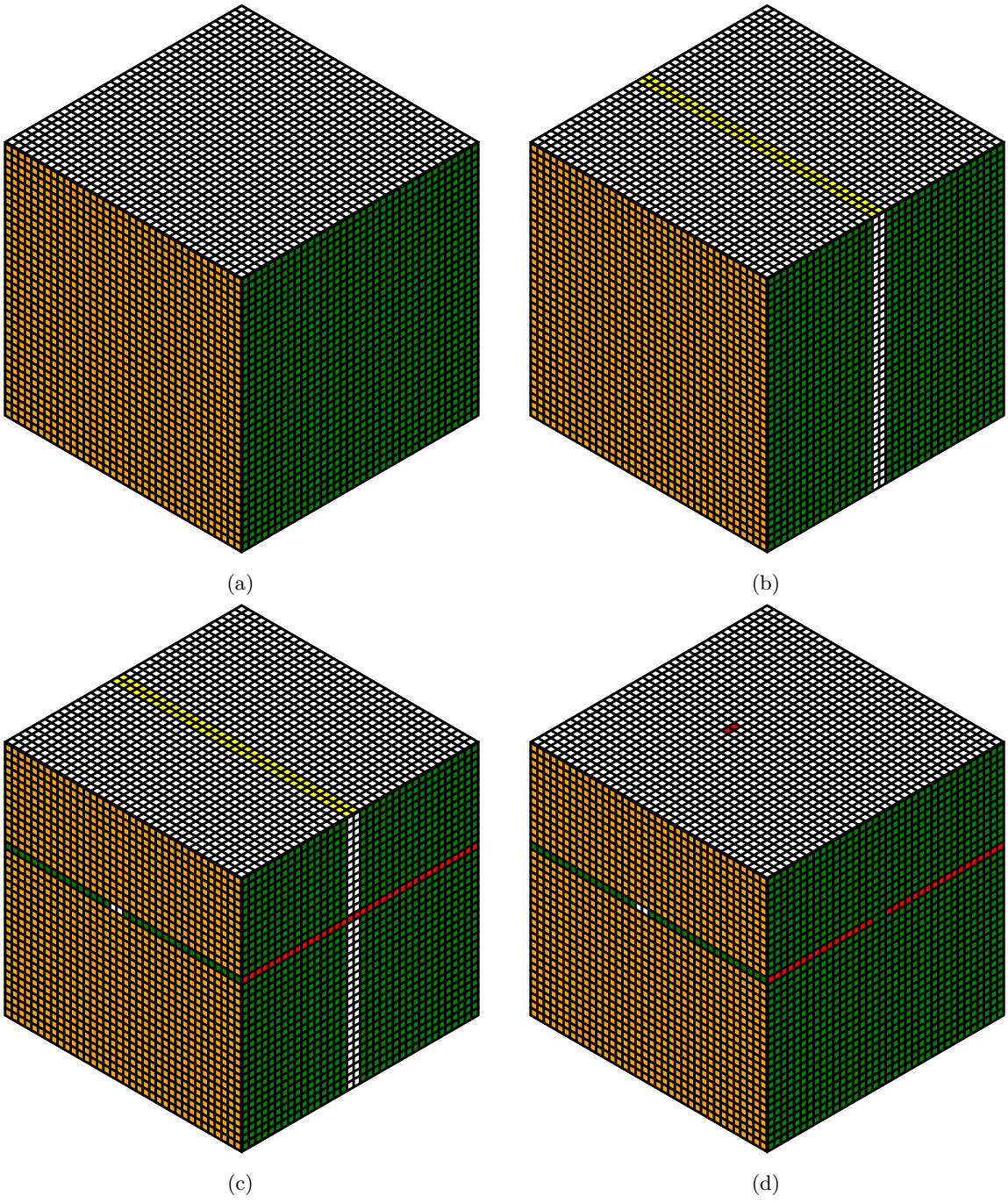
The proof of this theorem is involved and uninsightful. In addition, no other result from this section will be used in the rest of this chapter. As a result, the reader should feel free to skip the remainder of this section.

To formally derive the coloring of configuration C_b, we need to have a formal description of the effect of transformation b_i. For example, Figure 4-7 shows the $+x$, $+y$, and $+z$ faces of a Rubik's Cube in configurations C_0, $a_2(C_0)$, $(z_{m+2} \circ a_2)(C_0)$, and $b_2(C_0) = ((a_2)^{-1} \circ z_{m+2} \circ a_2)(C_0)$ where a_2 and $z_{m+2} = z_5$ are defined in terms of $l_2 = 110$ as in the reduction.

The exact behavior of a Rubik's Cube due to b_i is described by Lemmas 4.27 through 4.29:

Lemma 4.27. *Suppose $i \in \{1, \ldots, n\}$. Then the effect of b_i on the stickers from the $\pm z$ faces of a Rubik's Cube can be described as follows:*

- *If the jth bit of l_i is one, then the sticker starting on the $+z$ face with (x, y) coordinates $(j, -(m + i))$ ends up on the $+x$ face with (y, z) coordinates $(-j, (m + i))$.*



(a) (b) (c) (d)

Figure 4-7: Applying b_2 to C_0 step by step.

- *If the jth bit of l_i is one, then the sticker starting on the $-z$ face with (x, y) coordinates $(j, -(m+i))$ ends up on the $-x$ face with (y, z) coordinates $(-j, (m+i))$.*
- *All other stickers on the $\pm z$ faces stay in place.*

Proof. As noted in the proof of Lemma 4.23, a sticker is affected by $b_i = (a_i)^{-1} \circ z_{m+i} \circ a_i$ if and only if it is moved by the z_{m+i} term.

Consider the stickers originally on the $+z$ face. b_i starts with a_i, which rotates the x

slices with x coordinates j such that bit j of l_i is one. Therefore, the stickers on the $+z$ face with x coordinates of this form are rotated to the $+y$ face, and all the other stickers are left in place. After that, the only stickers from the $+z$ face which are moved by the z_{m+i} term of b_i are the stickers which were on the $+y$ face with z coordinate $(m+i)$ and x coordinate j such that bit j of l_i is one. In other words, the only stickers from the $+z$ face moved by the z_{m+i} term, are those starting at (x, y) coordinates $(j, -(m+i))$ where bit j of l_i is one.

All other stickers starting on the $+z$ face are not affected by the z_{m+i} term, and are therefore not moved by b_i. On the other hand, consider any sticker of this form: a sticker starting on the $+z$ face at (x, y) coordinates $(j, -(m+i))$ where bit j of l_i is one. Such a sticker is moved by a_i to (x, z) coordinates $(j, (m+i))$ of face $+y$. It is then moved by z_{m+i} to (y, z) coordinates $(-j, (m+i))$ of face $+x$. Finally, $(a_i)^{-1}$ does not affect the sticker since it is on the $+x$ face at the time and $(a_i)^{-1}$ consists of rotations of x slices.

Thus, if the jth bit of l_i is one, then the sticker starting on the $+z$ face with (x, y) coordinates $(j, -(m+i))$ ends up on the $+x$ face with (y, z) coordinates $(-j, (m+i))$. All other stickers starting on the $+z$ face remain in place.

The exact same logic applies to the stickers originally on the $-z$ face, allowing us to conclude that the lemma statement holds, as desired. □

Lemma 4.28. *Suppose $i \in \{1, \ldots, n\}$. Then the effect of b_i on the stickers from the $\pm y$ faces of a Rubik's Cube can be described as follows:*

- *If the jth bit of l_i does not exist (i.e. $j < 0$ or $j > m$) or if the jth bit of l_i is zero, then the sticker starting on the $+y$ face with (x, z) coordinates $(j, (m+i))$ ends up on the $+x$ face with (y, z) coordinates $(-j, (m+i))$.*
- *If the jth bit of l_i does not exist (i.e. $j < 0$ or $j > m$) or if the jth bit of l_i is zero, then the sticker starting on the $-y$ face with (x, z) coordinates $(j, (m+i))$ ends up on the $-x$ face with (y, z) coordinates $(-j, (m+i))$.*
- *All other stickers on the $\pm y$ faces stay in place.*

Proof. As noted in the proof of Lemma 4.23, a sticker is affected by $b_i = (a_i)^{-1} \circ z_{m+i} \circ a_i$ if and only if it is moved by the z_{m+i} term.

Consider the stickers originally on the $+y$ face. b_i starts with a_i, which rotates the x slices with x coordinates j such that bit j of l_i is one. Therefore, the stickers on the $+y$ face with x coordinates of this form are rotated to the $-z$ face, and all the other stickers are left in place. After that, the only stickers from the $+y$ face which are moved by the z_{m+i} term of b_i are the stickers with z coordinate $(m+i)$ which were not moved from the $+y$ face. In other words, the only stickers from the $+z$ face moved by the z_{m+i} term, are those starting at (x, y) coordinates $(j, -(m+i))$ where bit j of l_i either does not exist (i.e. $j < 0$ or $j > m$) or is zero.

All other stickers starting on the $+y$ face are not affected by the z_{m+i} term, and are therefore not moved by b_i. On the other hand, consider any sticker of this form: a sticker starting on the $+y$ face at (x, z) coordinates $(j, (m+i))$ where bit j of l_i either does not exist or is zero. Such a sticker is not moved by a_i. It is then moved by z_{m+i} to (y, z) coordinates $(-j, (m+i))$ of face $+x$. Finally, $(a_i)^{-1}$ does not affect the sticker since it is on the $+x$ face at the time and $(a_i)^{-1}$ consists of rotations of x slices.

Thus, if the jth bit of l_i does not exist (i.e. $j < 0$ or $j > m$) or if the jth bit of l_i is zero, then the sticker starting on the $+y$ face with (x, z) coordinates $(j, (m+i))$ ends up on

the $+x$ face with (y, z) coordinates $(-j, (m + i))$. All other stickers starting on the $+y$ face remain in place.

The exact same logic applies to the stickers originally on the $-y$ face, allowing us to conclude that the lemma statement holds, as desired. □

Lemma 4.29. *Suppose $i \in \{1, \ldots, n\}$. Then the effect of b_i on the stickers from the $\pm x$ faces of a Rubik's Cube can be described as follows:*

- *If the jth bit of l_i is one, then the sticker starting on the $+x$ face with (y, z) coordinates $(j, (m + i))$ ends up on the $-z$ face with (x, y) coordinates $(j, -(m + i))$.*
- *If the jth bit of l_i does not exist (i.e. $j < 0$ or $j > m$) or if the jth bit of l_i is zero, then the sticker starting on the $+x$ face with (y, z) coordinates $(j, (m + i))$ ends up on the $-y$ face with (x, z) coordinates $(j, (m + i))$.*
- *If the jth bit of l_i is one, then the sticker starting on the $-x$ face with (y, z) coordinates $(j, (m + i))$ ends up on the $+z$ face with (x, y) coordinates $(j, -(m + i))$.*
- *If the jth bit of l_i does not exist (i.e. $j < 0$ or $j > m$) or if the jth bit of l_i is zero, then the sticker starting on the $-x$ face with (y, z) coordinates $(j, (m + i))$ ends up on the $+y$ face with (x, z) coordinates $(j, (m + i))$.*
- *All other stickers on the $\pm x$ faces stay in place.*

Proof. As noted in the proof of Lemma 4.23, a sticker is affected by $b_i = (a_i)^{-1} \circ z_{m+i} \circ a_i$ if and only if it is moved by the z_{m+i} term.

Consider the stickers originally on the $+x$ face. b_i starts with a_i, which affects none of the stickers on the $+x$ face. After that, the z_{m+i} term moves exactly those stickers from the $+x$ face that had z coordinate $(m + i)$. As a result, these stickers are all affected by b_i, and all others are not.

Consider a sticker starting on the $+x$ face with (y, z) coordinates $(j, (m + i))$. This sticker is unaffected by a_i and then moved to the $-y$ face by z_{m+i}. In particular, it is moved to (x, z) coordinates $(j, (m + i))$. After that, there are two cases:

Case 1: If bit j of l_i does not exist (i.e. $j < 0$ or $j > m$) or if the jth bit of l_i is zero, then the sticker is unaffected by $(a_i)^{-1}$. This shows that if the jth bit of l_i does not exist or if the jth bit of l_i is zero, then the sticker starting on the $+x$ face with (y, z) coordinates $(j, (m + i))$ ends up on the $-y$ face with (x, z) coordinates $(j, (m + i))$.

Case 2: If bit j of l_i is one, then after being moved to the $-y$ face by z_{m+i}, the sticker in question is moved to the $-z$ face by $(a_i)^{-1}$. In particular, the sticker ends up at (x, y) coordinates $(j, -(m + i))$. This shows that if the jth bit of l_i is one, then the sticker starting on the $+x$ face with (y, z) coordinates $(j, (m + i))$ ends up on the $-z$ face with (x, y) coordinates $(j, -(m + i))$.

As previously mentioned, all stickers starting on the $+x$ face other than those addressed by the above cases stay in place due to b_i. Together with the statements shown in the two cases, this is exactly what we wished to show.

The same logic applies to the stickers originally on the $-x$ face, allowing us to conclude that the lemma statement holds, as desired. □

We can apply the above lemmas to figure out the effect of transformation $b_1 \circ b_2 \circ \cdots \circ b_n$ on configuration C_0. In particular, this allows us to learn the coloring of configuration C_b.

At this point, we can prove Theorem 4.26, which is restated below for convenience:

Theorem 4.26. *In C_b, the stickers have the following coloring:*

$+z$: The stickers on the $+z$ face with (x, y) coordinates $(j, -(m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all red. All other stickers are white.

$-z$: The stickers on the $-z$ face with (x, y) coordinates $(j, -(m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all orange. All other stickers are blue.

$+y$: The stickers on the $+y$ face with (x, z) coordinates $(j, (m + i))$ where $i \in \{1, \ldots, n\}$ and either l_i doesn't have a jth bit (i.e. $j < 0$ or $j > m$) or the jth bit of l_i is zero are all red. All other stickers are green.

$-y$: The stickers on the $-y$ face with (x, z) coordinates $(j, (m + i))$ where $i \in \{1, \ldots, n\}$ and either l_i doesn't have a jth bit (i.e. $j < 0$ or $j > m$) or the jth bit of l_i is zero are all orange. All other stickers are yellow.

$+x$: The stickers on the $+x$ face with (y, z) coordinates $(-j, (m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all white. All other stickers with z coordinate in $\{1, \ldots, n\}$ are green. All other stickers are orange.

$-x$: The stickers on the $-x$ face with (y, z) coordinates $(-j, (m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all blue. All other stickers with z coordinate in $\{1, \ldots, n\}$ are yellow. All other stickers are red.

Proof. C_b is obtained from C_0 by applying transformation $b_1 \circ b_2 \circ \cdots \circ b_n$. Each b_i affects a disjoint set of stickers. Using this fact together with the description of the effect of one b_i, we can obtain the description of the coloring of C_b given in the above theorem statement.

For example, consider the stickers that end up on the $+z$ face. According to Lemma 4.29, if the jth bit of l_i is one, then b_i moves the sticker starting on the $-x$ face with (y, z) coordinates $(-j, (m + i))$ to the $+z$ face with (x, y) coordinates $(j, -(m + i))$. Since the b_is each affect disjoint sets of stickers, the stickers on the $+z$ face with (x, y) coordinates $(j, -(m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all stickers that started on the $-x$ face. Since the $-x$ face is red in C_0, these stickers are all red. We know from Lemmas 4.27 through 4.29 that no stickers other than the ones that started there and those described above are moved to the $+z$ face by b_i. Therefore all other stickers on the $+z$ face started there. Since the $+z$ face is white in C_0, these stickers are all white. Putting this together, we obtain exactly the first bullet point of the theorem statement:

The stickers on the $+z$ face with (x, y) coordinates $(j, -(m + i))$ where $i \in \{1, \ldots, n\}$ and the jth bit of l_i is one are all red. All the other stickers are white.

The logic for the other five faces is exactly analogous, and is omitted here for brevity. □

This concludes the description of C_b in terms of colors. The coloring of configuration C_t—the configuration that is actually obtained by applying the reduction to $l_1, \ldots, l_n$—can be obtained from the coloring of configuration C_b by applying transformation a_1. This is shown for the previously given example in Figure 4-8.

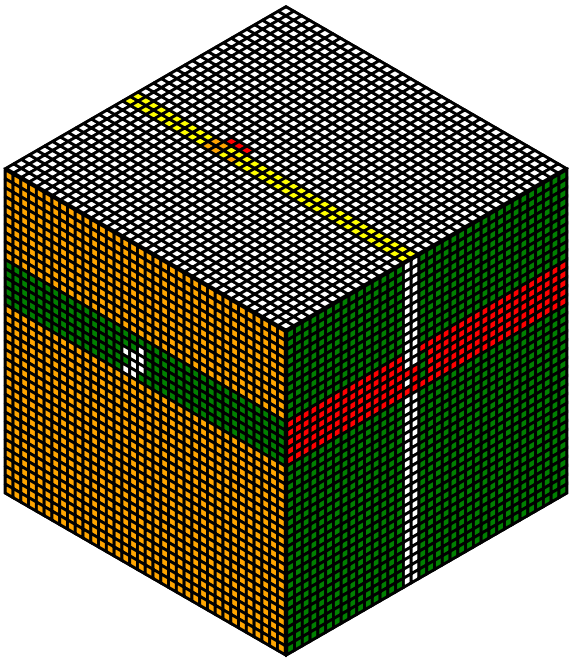


Figure 4-8: The $+x$, $+y$, and $+z$ faces of C_t for the example input $l_1, \ldots, l_n$.

4.5.4 (Group) STM/SQTM Rubik's Cube solution → Promise Cubical Hamiltonian Path solution: proof outline

We wish to prove the following:

Theorem 4.30. *If (C_t, k) is a "yes" instance to the STM Rubik's Cube problem, then $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

By Lemmas 4.6 and 4.7, this will immediately also imply the following corollary:

Corollary 4.31. *If (t, k) is a "yes" instance to the Group STM/SQTM Rubik's Cube problem or (C_t, k) is a "yes" instance to the STM/SQTM Rubik's Cube problem, then $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

The intuition behind the proof of this theorem is similar to that used in the Rubik's Square case, but there is added complexity due to the extra options available in a Rubik's Cube. Most of the added complexity is due to the possibility of face moves (allowing rows of stickers to align in several directions over the course of a solution).

Below, we describe an outline of the proof, including several high level steps, each of which is described in more detail in an additional subsection.

To prove the theorem, we consider a hypothetical solution to the (C_t, k) instance of the STM Rubik's Cube problem. A solution consists of a sequence of STM Rubik's Cube moves $m_1, \ldots, m_{k'}$ with $k' \leq k$ such that $C' = (m_{k'} \circ \cdots \circ m_1)(C_t)$ is a solved configuration of the Rubik's Cube.

One very helpful idea that is used several times throughout the proof is the idea of an index u such that no move m_i is an index-u move.

Definition 4.32. *Define $u \in \{m+n+1, m+n+2, \ldots, m+n+(2n)\}$ to be an index such that $m_1, \ldots, m_{k'}$ contains no index-u move.*

Notice that a value for u satisfying this definition must exist because variable u has $2n = k + 1 > k \geq k'$ possible values and each of the k' moves m_i disqualifies at most one possible value from being assigned to variable u.

Step 1 of the proof is a preliminary characterization of the possible index-$(m+i)$ moves among $m_1, \ldots, m_{k'}$ for $i \in \{1, \ldots, n\}$. Consider the following definition:

Definition 4.33. *Partition the set $\{1, \ldots, n\}$ into four sets of indices Z, O, T, and M (where Z, O, T, and M are named after "zero", "one", "two", and "more") as follows:*

- *$i \in Z$ if and only if $m_1, \ldots, m_{k'}$ contains exactly zero index-$(m+i)$ moves*
- *$i \in O$ if and only if $m_1, \ldots, m_{k'}$ contains exactly one index-$(m+i)$ move*
- *$i \in T$ if and only if $m_1, \ldots, m_{k'}$ contains exactly two index-$(m+i)$ moves*
- *$i \in M$ if and only if $m_1, \ldots, m_{k'}$ contains at least three index-$(m+i)$ moves*

In Step 1, we prove the following list of results, thereby restricting the set of possible index-$(m+i)$ moves (for $i \in \{1, \ldots, n\}$) among $m_1, \ldots, m_{k'}$:

- Z is empty.
- If $i \in O$, then the sole index-$(m+i)$ move must be a counterclockwise z turn.
- If $i \in T$, then the two index-$(m+i)$ moves must be a clockwise z turn and a z flip in some order.
- If $i \in O \cup T$, then any move of z slice $(m+i)$ must occur at a time when faces $+x$, $+y$, $-x$, and $-y$ all have zero rotation and any move of z slice $-(m+i)$ must occur at a time when these faces all have rotation $180°$.

Step 2 of the proof concerns the concept of paired stickers:

Definition 4.34. *Suppose p_1, p_2, and q are all distinct positive non-face slice indices. Then we say that two stickers are* (p_1, p_2, q)-paired *if the two stickers are on the same index-j slice, the two stickers are on the same quadrant of a face, one of the stickers has coordinates $\pm q$ and $\pm p_1$ within that face, and the second sticker has coordinates $\pm q$ and $\pm p_2$ within the face.*

In particular, we prove the following useful properties of paired stickers:

- If two stickers are (p_1, p_2, q)-paired, then they remain (p_1, p_2, q)-paired after one move unless the move is an index-p_1 move or an index-p_2 move which moves one of the stickers.
- Suppose $i_1, i_2 \in O$ and $j \in \{1, 2, \ldots, m\}$. Then consider any pair of stickers that are $(m + i_1, m + i_2, j)$-paired in C_b. If there are no face moves of faces $+x$, $+y$, $-x$, and $-y$ and no index-j moves that affect either of the stickers between the index-$(m + i_1)$ O-move and the index-$(m + i_2)$ O-move, then the two stickers remain $(m + i_1, m + i_2, j)$-paired in C'.

Step 3 of the proof uses a counting argument to significantly restrict the possible moves in $m_1, \ldots, m_{k'}$. In particular, consider the following classification of moves into disjoint types:

- "O-moves": index-$(m+i)$ moves with $i \in O$
- "T-moves": index-$(m+i)$ moves with $i \in T$
- "M-moves": index-$(m+i)$ moves with $i \in M$
- "J-moves": index-j moves with $j \in J = \{1, \ldots, m\}$
- "vertical face moves": face moves of faces $+x$, $+y$, $-x$, or $-y$
- "other moves": all other moves

We show using the results from Steps 1 and 2 that there must be a J-move or two vertical face moves between each pair of O-moves in $m_1, \ldots, m_{k'}$. As a result, we can count the number of moves of each type as follows:

Let c_O, c_T, c_M, $c_{vertical}$, c_J, and c_{other} be the number of moves of each type. We derive the following constraints:

- $c_O = |O|$
- $c_T = 2|T|$
- $c_M \geq 3|M|$
- $c_J + \frac{1}{2}c_{vertical} \geq |O| - 1$
- $c_{other} \geq 0$

Adding these together, we find that

$$k' - \frac{1}{2}c_{vertical} = c_O + c_T + c_M + \frac{1}{2}c_{vertical} + c_J + c_{other} \geq |O| + 2|T| + 3|M| + (|O| - 1) = k + |M|.$$

The above shows that $k' \geq k$, but we also know that $k' \leq k$. Thus, equality must hold at each step. Working out the details, we find that $c_O = |O|$, $c_T = 2|T|$, $c_J = |O| - 1$, and $c_M = c_{vertical} = c_{other} = 0$. Thus, the counting argument in this step shows that the only moves in $m_1, \ldots, m_{k'}$ other than O-moves and T-moves are the $|O| - 1$ quantity of J-moves which are between O-moves.

In Step 4, we further restrict the possibilities. In particular, we show the following:

- Since there are no face moves, the index-$(m+i)$ O-move for $i \in O$ can only be a counterclockwise z turn of slice $(m+i)$. Similarly the index-$(m+i)$ T-moves for $i \in T$ are a clockwise z turn and a z flip of slice $(m+i)$.
- Consider the elements $i \in O$ in the order in which their O-moves occur. We show that if i_1 is immediately before i_2 in this order, then it must be the case that l_{i_1} differs from l_{i_2} in exactly one bit.
- Furthermore, the one J-move between two consecutive O-moves of slices $m + i_1$ and $m + i_2$ must rotate the x slice whose index is the unique index j at which strings l_{i_1} and l_{i_2} differ.

At this point, we are almost done. Consider the elements $i \in O$ in the order in which their O-moves occur. The corresponding bitstring l_i in the same order have the property that each l_i is at Hamming distance one from the next. In Step 5, we use the ideas of paired stickers to show that T is empty, and as a result conclude that $O = \{1, \ldots, n\}$ and therefore that the above ordering of the l_is is an ordering of all the l_is in which each l_i has Hamming distance one from the next. In other words, we show our desired result: that $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.

4.5.5 Step 1: restricting the set of possible index-$(m+i)$ moves

As stated in the proof outline, we will prove the following list of results in this section

- Z is empty.
- If $i \in O$, then the sole index-$(m+i)$ move must be a counterclockwise z turn.
- If $i \in T$, then the two index-$(m+i)$ moves must be a clockwise z turn and a z flip in some order.
- If $i \in O \cup T$, then any move of z slice $(m+i)$ must occur at a time when faces $+x$, $+y$, $-x$, and $-y$ all have zero rotation and any move of z slice $-(m+i)$ must occur at a time when these faces all have rotation 180°.

We begin with a preliminary result concerning the coloring of the solved configuration $C' = (m_{k'} \circ \cdots \circ m_1)(C_t)$.

Lemma 4.35. *The solved Rubik's Cube configuration C' has the same face colors as C_0.*

Proof. Consider the sticker with both coordinates u on any face of C_0. No index-u moves occur within $m_{k'} \circ \cdots \circ m_1$ by definition of u. No index-u moves occur within $t = a_1 \circ b_1 \circ \cdots \circ b_n$ because t is defined entirely using moves of slices whose indices have absolute values at most $m+n$ and $u > m+n$. As a result, the sticker in question is never moved off of the face it starts on by the transformation $m_{k'} \circ \cdots \circ m_1 \circ t$. Applying transformation $m_{k'} \circ \cdots \circ m_1 \circ t$ to C_0 yields C', so the sticker is on the same face in C' as it is in C_0. Since both C_0 and C' are solved configurations, we conclude that configuration C' has the same face colors as C_0. □

Using this, we can show the first desired result:

Lemma 4.36. *Z is empty.*

Proof. Suppose for the sake of contradiction that $m_1, \ldots, m_{k'}$ contains no index-$(m+i)$ move for some $i \in \{1, \ldots, n\}$.

Then consider the sticker with coordinates $(x, z) = (u, m+i)$ on the $+y$ face of C_b. Configuration C' can be obtained from configuration C_b by transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1$. We know, however, that moves $m_1, \ldots, m_{k'}$ include no index-$(m+i)$ or index-u moves. Similarly, since $a_1 = (x_1)^{(l_1)_1} \circ \cdots \circ (x_m)^{(l_1)_m}$, we see that a_1 consists of no index-j moves with $j > m$. Since both $m+i$ and u are greater than m, we can conclude that transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1$ can be built without any index-$(m+i)$ or index-u moves. As a result, this transformation does not move the sticker in question to a different face.

We then see that the sticker with coordinates $(x, z) = (u, m + i)$ on the $+y$ face of C_b is also on the $+y$ face of C'. By Theorem 4.26, we see that the color of this sticker in C_b is red. However, the $+y$ face of C' is supposed to be the same color as the $+y$ face of C_0: green.

By contradiction, we see as desired that $m_1, \ldots, m_{k'}$ must contain some index-$(m + i)$ move for all $i \in \{1, \ldots, n\}$ □

The rest of what we wish to show concerns index-$(m + i)$ moves where $i \in O \cup T$.

For any i, we can restrict our attention to a specific set of stickers as in the following definition:

Definition 4.37. *Define the* special stickers *to be the* 48 *stickers in C_b with coordinates $\pm u$ and $\pm(m + i)$ (eight special stickers per face).*

Notice that by Theorem 4.26, all but 8 special stickers have the same color as the color of their starting face in C_0. This motivates the following further definition:

Definition 4.38. *Define the* correctly placed stickers *to be the* 40 *special stickers which have the same color as their starting face has in C_0. Define the* misplaced stickers *to be the other* 8 *special stickers.*

Of the 8 misplaced stickers, the two on the $+y$ face have the color of the $-x$ face in C_0, the two on the $-x$ face have the color of the $-y$ face in C_0, the two on the $-y$ face have the color of the $+x$ face in C_0, and the two on the $+x$ face have the color of the $+y$ face in C_0. In short, starting at C_b, the 8 misplaced stickers must each move one face counterclockwise around the z axis in order to end up on the face whose color in C_0 matches the color of the sticker.

Next, consider the effect that move sequence $m_1, \ldots, m_{k'}$ must have on the special stickers

Lemma 4.39. *When starting in configuration C_b, move sequence $m_1, \ldots, m_{k'}$ must move the misplaced stickers one face counterclockwise around the z axis and must return each of the correctly placed stickers to the face that sticker started on.*

Proof. Configuration C', which has the same coloring scheme as configuration C_0, can be reached from configuration C_b by applying transformation $m_{k'} \circ \cdots \circ m_1 \circ a_1$. Therefore, the 8 misplaced stickers must be moved counterclockwise one face around the z axis and the 40 correctly placed stickers must stay on the same face due to this transformation. Notice that the only moves which transfer special stickers between faces are index-u and index-$(m + i)$ moves. The only other moves that even affect special stickers are face moves. As previously argued, a_1 contains no index-u moves. In fact, a_1 does not contain face moves or index-$(m + i)$ moves either and so a_1 does not move any of the special stickers at all.

In other words, the effect of this transformation $(m_{k'} \circ \cdots \circ m_1 \circ a_1)$ on the special stickers is the same as the effect of just the transformation $m_1, \ldots, m_{k'}$. Thus, $m_1, \ldots, m_{k'}$ must move the misplaced stickers one face counterclockwise around the z axis and must return each of the correctly placed stickers to the face that sticker started on. □

This allows us to directly prove the next two parts of our desired result:

Lemma 4.40. *If $i \in O$, then the sole index-$(m + i)$ move must be a counterclockwise z turn.*

Proof. Consider the result of move sequence $m_1, \ldots, m_{k'}$ when starting in configuration C_b. We showed above that the 8 misplaced stickers must each move one face counterclockwise around the z axis and the correctly placed stickers must stay on the same face. Furthermore, the only moves which cause special stickers to change faces are index-u or index-$(m+i)$ moves. Since $m_1, \ldots, m_{k'}$ includes no index-u moves and includes exactly one index-$(m+i)$ move (for $i \in O$), we see that the special stickers only change faces during the sole index-$(m+i)$ move in $m_1, \ldots, m_{k'}$.

Every slice with index $\pm(m+i)$ contains exactly 8 special stickers; therefore the sole index-$(m+i)$ move must cause exactly 8 of the special stickers to change faces.

In order for the 8 misplaced stickers to change faces and for the correctly placed stickers not to, it must be the case that the single index-$(m+i)$ move relocates exactly the 8 misplaced stickers. These stickers are on the $\pm x$ and $\pm y$ faces. Since the single index-$(m+i)$ move affects 8 stickers on the $\pm x$ and $\pm y$ faces and moves each of these stickers exactly one face counterclockwise around the z axis, it must be the case that this move is a counterclockwise z slice turn. As desired, the sole index-$(m+i)$ move is a counterclockwise z turn. □

Lemma 4.41. *If $i \in T$, then the two index-$(m+i)$ moves must be a clockwise z turn and a z flip in some order.*

Proof. As before, consider the result of move sequence $m_1, \ldots, m_{k'}$ when starting in configuration C_b. The 8 misplaced stickers must each move one face counterclockwise around the z axis and the correctly placed stickers must stay on the same face. Since the only moves which cause special stickers to change faces are index-u or index-$(m+i)$ moves, the only moves among $m_1, \ldots, m_{k'}$ which move special stickers between faces are the two index-$(m+i)$ moves (for $i \in T$).

Notice that every slice with index $\pm(m+i)$ contains exactly 8 special stickers, so each of the two index-$(m+i)$ moves must cause exactly 8 of the special stickers to change faces.

We proceed by casework:

- If exactly one of the two index-$(m+i)$ moves is an x or y move, then at least one of the correctly placed stickers from the $+z$ face is moved from that face and never returned there. Note that correctly placed stickers are supposed to end up on their starting faces.

- If both index-$(m+i)$ moves are x moves, then the misplaced stickers from the $+y$ face never leave that face. Note that misplaced stickers are supposed to move from their starting faces.

- If both index index-$(m+i)$ moves are y moves, then the misplaced stickers from the $+x$ face never leave that face. Note that misplaced stickers are supposed to move from their starting faces.

- If the first index-$(m+i)$ move is an x move and the second is a y move, then each of the misplaced stickers from the $+x$ face end up on the $\pm x$ or $\pm z$ faces. Note that misplaced stickers from the $+x$ face are supposed to move to the $+y$ face.

- If the first index-$(m+i)$ move is a y move and the second is an x move, then each of the misplaced stickers from the $+y$ face end up on the $\pm y$ or $\pm z$ faces. Note that misplaced stickers from the $+y$ face are supposed to move to the $-x$ face.

Since all these cases lead to contradiction, we can conclude that the only remaining case holds: both index-$(m+i)$ moves must be z moves.

Next suppose for the sake of contradiction that the 8 special stickers which are moved by one index-$(m+i)$ move are not the same as the special stickers moved by the other index-$(m+i)$ move. Any special sticker moved by exactly one of these moves will change faces and must therefore be a misplaced sticker. That sticker must move one face counterclockwise around the z axis. Since each of the two index-$(m+i)$ moves includes at least one sticker that is not moved by the other index-$(m+i)$ move we can conclude that the two index-$(m+i)$ moves are both counterclockwise z turns. Then any sticker moved by both index-$(m+i)$ moves is moved two faces counterclockwise around the z axis. This is not the desired behavior for any of the special stickers so none of the stickers can be moved by both index-$(m+i)$ moves. Thus there are a total of 16 different special stickers, each of which is moved by exactly one of the two index-$(m+i)$ moves. All 16 of these stickers end up on a different face from the one they started at. This is a contradiction since there are only 8 misplaced stickers.

We conclude that the two moves affect the same 8 stickers. The only way to rotate a total of one quarter rotation counterclockwise with two moves is using one clockwise turn and one flip. Thus, as desired, the two index-$(m+i)$ moves for $i \in T$ must be a clockwise z turn and a z flip in some order. □

Finally, we have only one thing left to prove in this section:

Lemma 4.42. *If $i \in O \cup T$, then any move of z slice $(m+i)$ must occur at a time when faces $+x$, $+y$, $-x$, and $-y$ all have zero rotation and any move of z slice $-(m+i)$ must occur at a time when faces $+x$, $+y$, $-x$, and $-y$ all have rotation $180°$.*

Proof. As before, consider the result of move sequence $m_1, \ldots, m_{k'}$ when starting in configuration C_b. The 8 misplaced stickers must each move one face counterclockwise around the z axis and the correctly placed stickers must stay on the same face. The only moves which cause special stickers to change faces are index-u or index-$(m+i)$ moves, though face moves also move special stickers. The only moves among $m_1, \ldots, m_{k'}$ which move special stickers between faces are the one or two index-$(m+i)$ moves (for $i \in O \cup T$). Furthermore, as shown in the proofs of Lemmas 4.40 and 4.41, the special stickers which are affected by these moves are exactly the misplaced stickers. In other words, throughout the entire move sequence $m_1, \ldots, m_{k'}$, the only moves which affect the correctly placed stickers are the face moves.

Let m_j be any index-$(m+i)$ move. According to Lemmas 4.40 and 4.41, m_j rotates a z slice.

Consider the six correctly placed stickers on one of the $\pm x$ or $\pm y$ faces. Since these stickers are only ever affected by face moves, their coordinates within the face are completely determined by the total rotation of the face so far. If the total rotation so far is 0, then the six correctly placed stickers are in the positions with coordinates $\pm u$ and $\pm(m+i)$ and with $z \neq (m+i)$. If the total rotation so far is $90°$, then the six correctly placed stickers are in the positions with coordinates $\pm u$ and $\pm(m+i)$ and with $x \neq -(m+i)$ for the $\pm y$ faces or $y \neq (m+i)$ for the $\pm x$ faces. If the total rotation so far is $180°$, then the six correctly placed stickers are in the positions with coordinates $\pm u$ and $\pm(m+i)$ and with $z \neq -(m+i)$. If the total rotation so far is $270°$, then the six correctly placed stickers are in the positions with coordinates $\pm u$ and $\pm(m+i)$ and with $x \neq (m+i)$ for the $\pm y$ faces or $y \neq -(m+i)$ for the $\pm x$ faces.

The only way for move m_j to avoid affecting these stickers if m_j rotates z slice $(m+i)$ is for the stickers to be in the positions with $z \neq (m+i)$. In other words, the total rotation of the face must be 0. The only way for move m_j to avoid affecting these stickers if m_j rotates z slice $-(m+i)$ is for the stickers to be in the positions with $z \neq -(m+i)$. In other words, the total rotation of the face must be 180°. This logic applies to each of the $\pm x$ and $\pm y$ faces. In other words, if m_j is some move with index $(m+i)$, then each of the $\pm x$ and $\pm y$ faces must have rotation 0 and if m_j is some move with index $-(m+i)$, then each of the $\pm x$ and $\pm y$ faces must have rotation 180°. □

4.5.6 Step 2: exploring properties of paired stickers

As stated in the proof outline, this step of the proof explores the properties of paired stickers.

Lemma 4.43. *If two stickers are (p_1, p_2, q)-paired, then they remain (p_1, p_2, q)-paired after one move unless the move is an index-p_1 move or an index-p_2 move which moves one of the stickers.*

Proof. Consider the effect of any move on the two stickers.

If the move doesn't affect either sticker, then the two stickers maintain their coordinates (and therefore also stay on the same face quadrant and slice). Thus the two stickers remain (p_1, p_2, q)-paired.

If the move moves both stickers, then they both rotate the same amount. In other words, as far as those two stickers are concerned, the effect of the move is the same as the effect of rotating the entire Rubik's Cube. When rotating the Rubik's Cube, two stickers sharing a slice continue to share a slice, two stickers sharing a face quadrant continue to share a face quadrant, and each sticker maintains the same set of coordinate absolute values as it had before. Thus, the two stickers remain (p_1, p_2, q)-paired.

Clearly, the only way for the stickers to no longer be (p_1, p_2, q)-paired is for the move to affect exactly one of the stickers. The possible moves affecting the stickers are face moves, index-q moves, index-p_1 moves, and index-p_2 moves. Among these, face moves and index-q moves necessarily affect either both stickers in the pair or neither. Thus, the only way for the stickers to stop being (p_1, p_2, q)-paired is via an index-p_1 move or an index-p_2 move which moves one of the stickers. □

Using this, we prove the following lemma:

Lemma 4.44. *Suppose $i_1, i_2 \in O$ and $j \in \{1, 2, \ldots, m\}$. Then consider any pair of stickers that are $(m+i_1, m+i_2, j)$-paired in C_b. If there are no face moves of faces $+x$, $+y$, $-x$, and $-y$ and no index-j moves that affect either of the stickers between the index-$(m+i_1)$ O-move and the index-$(m+i_2)$ O-move, then the two stickers remain $(m+i_1, m+i_2, j)$-paired in C'.*

Proof. Consider two $(m+i_1, m+i_2, j)$-paired stickers in C_b. Suppose that there exists neither an index-j move affecting one of the stickers nor a face move of face $+x$, $+y$, $-x$, or $-y$ between the index-$(m+i_1)$ and index-$(m+i_2)$ O-moves. Let m_α be the index-$(m+i_1)$ O-move and let m_β be the index-$(m+i_2)$ O-move. Without loss of generality, suppose m_α occurs before m_β.

Since there are no $+x$, $+y$, $-x$, or $-y$ face moves between m_α and m_β, we know that the rotations of these faces remain the same at the times of both moves. Applying the results from Step 1, either m_α and m_β are both counterclockwise turns of z slices $(m+i_1)$ and $(m+i_2)$ or m_α and m_β are counterclockwise turns of z slices $-(m+i_1)$ and $-(m+i_2)$.

Configuration C' can be obtained from configuration C_b by applying transformation $m_{k'} \circ \cdots \circ m_2 \circ m_1 \circ a_1$. Since a_1 consists of some number of x-slice turns, we can represent this transformation as a sequence of moves. We know that since the stickers are $(m+i_1, m+i_2, j)$-paired in C_b, they must remain $(m+i_1, m+i_2, j)$-paired until immediately before the first index-$(m+i_1)$ or index-$(m+i_2)$ move: m_α. We will show below that because of our assumption, the stickers will also end up $(m+i_1, m+i_2, j)$-paired immediately after m_β in all cases.

The first case is that the stickers are on face $+z$ or face $-z$ immediately before m_α. In that case, move m_α, which is a z move, will not affect either sticker. As a result, the two stickers will remain $(m+i_1, m+i_2, j)$-paired after m_α. With the exception of m_α and m_β, the only moves in $m_1, \ldots, m_{k'}$ which move these two stickers between faces are index-j moves. But by assumption, there are no index-j moves occurring between m_α and m_β which affect the stickers. Thus, immediately before m_β, the two stickers will still be $(m+i_1, m+i_2, j)$-paired and will still be on face $+z$ or face $-z$. As a result, m_β will also not affect the stickers. Therefore, they will remain $(m+i_1, m+i_2, j)$-paired immediately after m_β.

The second case is that the stickers are on face $+x$, $+y$, $-x$, or $-y$ immediately before m_α. Between m_α and m_β, the only moves in $m_1, \ldots, m_{k'}$ which move these two stickers are face moves and index-j moves. No matter how m_α affects the two stickers, they will both remain on the four faces $+x$, $+y$, $-x$, and $-y$. By assumption, neither sticker will be moved by an index-j move until m_β. Since the stickers are on faces $+x$, $+y$, $-x$, or $-y$, the assumption tells us that neither sticker will be moved by a face move until m_β either. Thus, the next move after m_α which affects either sticker is m_β.

Immediately before m_α, the first sticker has z coordinate $(m+i_1)$ if and only if the second sticker has z coordinate $(m+i_2)$. Similarly, the first sticker has z coordinate $-(m+i_1)$ if and only if the second sticker has z coordinate $-(m+i_2)$. This is simply a consequence of the definition of $(m+i_1, m+i_2, j)$-paired stickers. We know that m_α and m_β together either rotate z slices $(m+i_1)$ and $(m+i_2)$ counterclockwise one turn or rotate z slices $-(m+i_1)$ and $-(m+i_2)$ counterclockwise one turn. Thus in any case we see that over the course of the moves from m_α to m_β, either both stickers are rotated counterclockwise one turn around the z axis or neither is. As far as the two stickers are concerned, that is equivalent to a rotation of the entire Rubik's Cube. That means that in this case as well, the two stickers remain $(m+i_1, m+i_2, j)$-paired immediately after m_β.

We see that in both cases the two stickers remain $(m+i_1, m+i_2, j)$-paired immediately after m_β. Since there are no index-$(m+i_1)$ or index-$(m+i_2)$ moves after m_β, we know that the two stickers will continue to be $(m+i_1, m+i_2, j)$-paired until C'. □

4.5.7 Step 3: classifying possible moves with a counting argument

As stated in the proof outline, this step uses a counting argument to restrict the possible moves in $m_1, \ldots, m_{k'}$.

To begin, we show the following:

Lemma 4.45. *There must be a J-move or two vertical face moves between each pair of O-moves in $m_1, \ldots, m_{k'}$.*

Proof. Consider any pair of O-moves m_α and m_β which occur in that order. Suppose m_α is an index-$(m+i_1)$ move and m_β is an index-$(m+i_2)$ move. Let j be an index such that $(l_{i_1})_j$ differs from $(l_{i_2})_j$.

Notice that the $(m+i_1, m+i_2, j)$-paired stickers on face $+y$ with (x,z) coordinates $(j, m+i_1)$ and $(j, m+i_2)$ have different colors in C_b (see Theorem 4.26). Therefore they cannot be $(m+i_1, m+i_2, j)$-paired in C'. By the contraposative of Lemma 4.44, we see that at least one index-j move affecting one of these stickers or at least one face move of faces $+x$, $+y$, $-x$, or $-y$ must occur between m_α and m_β.

We know from the results of Step 1 that at the times of m_α and m_β, the four faces $\pm x$ and $\pm y$ must either each have total rotation of 0 or total rotation of 180°. Thus between the two moves, either the rotations of all four faces must change, or the rotation of any face that changes must also change back. Therefore it is impossible for exactly one face move of faces $+x$, $+y$, $-x$, and $-y$ to occur between these two moves.

In other words, we have shown that at least one J-move or at least two vertical face moves must occur between m_α and m_β. □

Corollary 4.46. *If*

- *m_α and m_β are index-$(m+i_1)$ and index-$(m+i_2)$ O-moves,*
- *l_{i_1} and l_{i_2} differ in bit j, and*
- *there are no vertical face moves between m_α and m_β,*

then there must be an index-j J-move between m_α and m_β.

Proof. This follows directly from one of the cases in the previous proof. □

With that done, we can count the number of moves of each type as follows:

There is exactly one O-move for each $i \in O$ (the sole index-$(m+i)$ move), so therefore $c_O = |O|$.

There are exactly two T-move for each $i \in T$ (the two index-$(m+i)$ moves), so therefore $c_T = 2|T|$.

There are at least three M-moves for each $i \in M$ (the index-$(m+i)$ moves), so therefore $c_M \geq 3|M|$.

Consider the O-moves in order. Between the $c_O = |O|$ different O-moves there are $|O|-1$ gaps. As shown above, each such gap must contain either at least one J-move or at least two vertical face moves. Therefore the number of J-moves plus half the number of vertical face moves upper-bounds the number of gaps: $c_J + \frac{1}{2}c_{vertical} \geq |O| - 1$.

Finally, $c_{other} \geq 0$.

Putting this together, we see the following:

$$\begin{aligned}
k' &= c_O + c_T + c_M + c_{vertical} + c_J + c_{other} \\
&= c_O + c_T + c_M + \left(c_J + \frac{1}{2}c_{vertical}\right) + \frac{1}{2}c_{vertical} + c_{other} \\
&\geq |O| + 2|T| + 3|M| + (|O| - 1) + \frac{1}{2}c_{vertical} \\
&= 2|O| + 2|T| + 3|M| - 1 + \frac{1}{2}c_{vertical} \\
&= 2n - 1 + |M| + \frac{1}{2}c_{vertical} \\
&= k + |M| + \frac{1}{2}c_{vertical} \\
&\geq k
\end{aligned}$$

The above shows that $k' \geq k$, but we also know that $k' \leq k$. Thus, equality must hold at each step. In particular, c_M must equal $3|M|$, $\left(c_J + \frac{1}{2}c_{vertical}\right)$ must equal $|O| - 1$, c_{other} must equal 0, and $|M| + \frac{1}{2}c_{vertical}$ must equal 0.

Since $|M| + \frac{1}{2}c_{vertical} = 0$, we can conclude that both $|M|$ and $c_{vertical}$ are equal to 0. Thus $c_M = 3|M| = 0$ also holds. All together, this shows that $c_O = |O|$, $c_T = 2|T|$, $c_J = |O| - 1$, and $c_M = c_{vertical} = c_{other} = 0$.

4.5.8 Step 4: further restricting possible move types

As stated in the proof outline, we will prove the following list of results in this section:

- Since there are no face moves, the index-$(m+i)$ O-move for $i \in O$ can only be a counterclockwise z turn of slice $(m+i)$. Similarly the index-$(m+i)$ T-moves for $i \in T$ are a clockwise z turn and a z flip of slice $(m+i)$.
- Consider the elements $i \in O$ in the order in which their O-moves occur. We show that if i_1 is immediately before i_2 in this order, then it must be the case that l_{i_1} differs from l_{i_2} in exactly one bit.
- Furthermore, the one J-move between two consecutive O-moves of slices $m+i_1$ and $m+i_2$ must rotate the x slice whose index is the unique index j at which strings l_{i_1} and l_{i_2} differ.

Lemma 4.47. *If $i \in O$, the single index-$(m+i)$ move in $m_1, \ldots, m_{k'}$ is a counterclockwise z turn of slice $(m+i)$.*

Proof. We have already seen that the move in question must be either a counterclockwise z turn of slice $(m+i)$ or a counterclockwise z turn of slice $-(m+i)$. Furthermore, the slice being rotated is slice $(m+i)$ if at the time of the move each vertical face ($\pm x$ and $\pm y$) has the total rotation 0. We have already seen, however, that none of the moves in $m_1, \ldots, m_{k'}$ are face moves. Thus the total rotation of each face is always 0, and as desired, the move in question is a counterclockwise z turn of slice $(m+i)$. □

Lemma 4.48. *If $i \in T$, the two index-$(m+i)$ moves in $m_1, \ldots, m_{k'}$ are a clockwise z turn of slice $(m+i)$ and a z flip of slice $(m+i)$.*

Proof. This proof follows analagously to the previous. □

Lemma 4.49. *Suppose that m_α and m_β are two O-moves of slices $(m+i_1)$ and $(m+i_2)$ with no other O-moves between them. It must be the case that l_{i_1} differs from l_{i_2} in exactly one bit.*

Proof. We have already seen that there must be at least one J-move between m_α and m_β. In fact, there has to be exactly one J-move in each of the $|O| - 1$ "gaps" between O-moves, so there can only be one J-move between m_α and m_β.

We saw in Corollary 4.46, however, that if l_{i_1} and l_{i_2} differ in bit j, then there must be an index-j J-move between m_α and m_β. As desired, we conclude that l_{i_1} and l_{i_2} must differ in at most one bit j. Since the bitstrings are all distinct, this is exactly what we were trying to show. □

Lemma 4.50. *Suppose that m_α and m_β are two O-moves of slices $(m+i_1)$ and $(m+i_2)$ with no other O-moves between them. If l_{i_1} differs from l_{i_2} in bit j, then it must be the case that the one J-move between m_α and m_β must rotate the x slice with index j.*

Proof. We know that the J-move in question must rotate a slice with index $\pm j$. We want to show that the move rotates the x slice with index j in particular.

Consider the pair of stickers in C_b at (x, z) coordinates $(j, m+i_1)$ and $(j, m+i_2)$ on the $+y$ face and also the pair of stickers in C_b at (x, y) coordinates $(j, -(m+i_1))$ and $(j, -(m+i_2))$ on the $+z$ face. These two pairs of stickers are both $(m+i_1, m+i_2, j)$-paired. Furthermore, each of these two pairs contain stickers of two different colors (see Theorem 4.26).

To transition from C_b to C', we apply transformation $m_k \circ \cdots \circ m_1 \circ a_1$. In other words, we apply a sequence of moves starting with some number of x turns (making up a_1) and then proceeding through move sequence $m_1, \ldots, m_k$. Because the solution contains no face moves, the only moves in this list before m_α which affect the four stickers in question are rotations of the x slice with index j. No matter how much or how little this slice rotates, one of the two pairs of stickers will be on face $+z$ or $-z$.

Consider that pair. Move m_α will be a counterclockwise z turn and therefore will not affect either sticker in the pair. That pair of stickers cannot be $(m+i_1, m+i_2, j)$-paired in C' since they have different colors. Since m_β is the only other index-$(m+i_1)$ or index-$(m+i_2)$ move, we can conclude from Lemma 4.43 that one of the two stickers must be affected by m_β. In order for that to be the case, however, the sole J-move between m_α and m_β must move the stickers in this pair off of the $\pm z$ face. Notice that the J-move between m_α and m_β must rotate a slice with index $\pm j$. Since there are no face moves in the solution, the only option which meets the requirements is to have the J-move rotate the x slice with index j. □

4.5.9 Step 5: showing T is empty

As stated in the proof outline, the purpose of this step is to show that T is empty. That on its own is sufficient to complete the proof.

Lemma 4.51. *When applying the move sequence $a_1, m_1, \ldots, m_k$ to C_b, the stickers with $z = i$ and $1 \le x \le n$ of face $+y$ for $i \in O$ immediately after the O-move of slice $(m+i)$ are the ones which started in the corresponding positions $z = i$ and $1 \le -y \le n$ of the face $+x$ in C_b.*

Proof. Let m_α be the O-move of slice $(m+i)$.

Consider the stickers in positions $z = i$ and $1 \le x \le n$ of face $+y$ for $i \in O$ immediately after the move m_α. These stickers were moved there by m_α from positions $z = i$ and $1 \le -y \le n$ of the face $+x$.

All O-moves and T-moves prior to m_α affect z slices whose indices are not i. All J-moves and all moves comprising a_1 affect non-face x slices and therefore don't affect the $+x$ face. As a result, no move in $a_1, m_1, \ldots, m_k$ before m_α affects the stickers with $z = i$ and $-n \le y \le -1$ on the $+x$ face. Thus, the stickers in positions $z = i$ and $1 \le -y \le n$ of the face $+x$ immediately before m_α are the same as the stickers in those positions in configuration C_b.

As desired, the stickers with $z = i$ and $1 \le x \le n$ of face $+y$ for $i \in O$ immediately after the move m_α are the ones which started in the corresponding positions $z = i$ and $1 \le -y \le n$ of the face $+x$ in C_b. □

Lemma 4.52. *When applying the move sequence $a_1, m_1, \ldots, m_k$ to C_b, the stickers with $z = i$ and $1 \leq x \leq n$ of face $+y$ for $i \in T$ after the second T-move rotating a slice with index $(m + i)$ are the ones which started in the corresponding positions $z = i$ and $1 \leq -y \leq n$ of the face $+x$ in C_b.*

Proof. Let m_α and m_β be the two T-moves of slice $(m + i)$.

Consider the stickers in positions $z = i$ and $1 \leq x \leq n$ of face $+y$ immediately after m_β. These stickers were moved there by m_β either from positions $z = i$ and $1 \leq y \leq n$ of the face $-x$ or from positions $z = i$ and $1 \leq -x \leq n$ of face $-y$ (depending on whether the second T-move is the turn or the flip).

In either case, none of the moves between m_α and m_β could have affected any of these stickers (since the moves in that interval are all either O- or T- moves moving z slices of other indices or J-moves moving x slices with indices 1 through n). Therefore immediately before m_α, these stickers were in positions $z = i$ and $1 \leq -y \leq n$ of the face $+x$. Once again, no moves before that could affect these stickers, so these stickers must have started in that position in C_b.

As desired, the stickers with $z = i$ and $1 \leq x \leq n$ of face $+y$ immediately after the move m_β are the ones which started in the corresponding positions $z = i$ and $1 \leq -y \leq n$ of the face $+x$ in C_b. □

Theorem 4.53. *T is empty.*

Proof. Note that O cannot be empty since then the number of J-moves would be $|O|-1 = -1$.

Suppose for the sake of contradiction that $i_1 \in T$. Consider the second T-move of the z slice with index $(m + i_1)$ in move sequence $a_1, m_1, \ldots, m_k$. Call this move m_α. The move m_α cannot be seperated from every O-move by J-moves because if that were the case, there would be two J-moves without an O-move between them (or in other words there would be two O-moves with at least two J-moves between them). Thus there must be some O-move m_β of slice $(m + i_2)$ that is not seperated from m_α by any J-move.

Consider what happens if we apply the move sequence $a_1, m_1, \ldots, m_k$ to C_b until right after both m_α and m_β have occurred. Call this configuration C_{mid}. For every $j \in \{1, \ldots, m\}$, the stickers that are in (x, z) coordinates $(j, m + i_1)$ and $(j, m + i_2)$ of face $+y$ in C_{mid} are $(m + i_1, m + i_2, j)$-paired. When transitioning from C_{mid} to C', no index-$(m + i_1)$ or index-$(m + i_2)$ moves occur, and so these stickers are also $(m + i_1, m + i_2, j)$-paired in C'. Thus we conclude that the stickers in each pair are the same color.

Therefore we have that in C_{mid}, the stickers on face $+y$ with $z = i_2$ and $1 \leq x \leq n$ have the same color scheme, call it S, as the stickers on face $+y$ with $z = i_1$ and $1 \leq x \leq n$. Before we reach the configuration C_{mid}, the final few moves are a sequence of O-moves and T-moves including m_α and m_β. Furthermore, among these O-moves and T-moves, none that occur after m_α affect the stickers with $z = i_1$ and none that occur after m_β affect the stickers with $z = i_2$. Therefore the color scheme of the stickers in positions $z = i_2$ and $1 \leq x \leq n$ of face $+y$ immediately after m_β is the same as S: the color scheme of those stickers in C_{mid}. Similarly, the color scheme of the stickers in positions $z = i_1$ and $1 \leq x \leq n$ of face $+y$ immediately after m_α is also S. Using Lemmas 4.51 and 4.52, we conclude that the color scheme of the stickers in positions $z = i_2$ and $1 \leq -y \leq n$ of face $+x$ in configuration C_b is S and that the color scheme of the stickers in positions $z = i_1$ and $1 \leq -y \leq n$ of face $+x$ in configuration C_b is also S. This, however, is a contradiction, since those two color schemes in C_b are different for any two different i_1 and i_2 (see Theorem 4.26).

We conclude that $i_1 \in T$ cannot exist, and therefore that T is empty. □

This completes the proof of Theorem 4.30 outlined in Section 4.5.4.

4.5.10 Conclusion

Theorems 4.24 and 4.30 and Corollaries 4.25 and 4.31 show that the polynomial time reductions given are answer preserving. As a result, we conclude that

Theorem 4.54. *The STM/SQTM Rubik's Cube and Group STM/SQTM Rubik's Cube problems are NP-complete.*

4.6 Future work

In this chapter, we resolve the complexity of optimally solving Rubik's Cubes under move count metrics for which a single move rotates a single slice. It could be interesting to consider the complexity of this problem under other move count metrics.

Of particular interest are the Wide Turn Metric (WTM) and Wide Quarter Turn Metric (WQTM), in which the puzzle solver can rotate any number of contiguous layers together provided they include one of the faces. These move count metrics are the closest to how one would physically solve a real-world $n \times n \times n$ Rubik's Cube: by grabbing some of the slices in the cube (including a face) from the side and rotating those slices together. We can also consider the $1 \times n \times n$ analogue of the Rubik's Cube with WTM move count metric: this would be a Rubik's Square in which a single move flips a contiguous sequence of rows or columns including a row or column at the edge of the Square. Solving this toy model could help point us in the right direction for the WTM and WQTM Rubik's Cube problems. If even the toy model resists analysis, it could be interesting to consider this toy model with missing stickers.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Daniel Andersson. Hashiwokakero is NP-complete. *Information Processing Letters*, 109(19):1145–1146, 2009.

[2] Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.

[3] Esther M. Arkin, Sándor P. Fekete, Kamrul Islam, Henk Meijer, Joseph S. B. Mitchell, Yurai Núñez-Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super)thin or solid is hard: A study of grid Hamiltonicity. *Computational Geometry: Theory and Applications*, 42(6–7):582–605, 2009. Originally published at EuroComb'07.

[4] Esther M. Arkin, Sándor P. Fekete, and Joseph S.B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry: Theory and Applications*, 17(1–2):25–50, 2000.

[5] Mark Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In *Proceedings of the 16th Annual International Conference on Computing and Combinatorics*, volume 6196 of *Lecture Notes in Computer Science*, pages 216–225, Nha Trang, Vietnam, July 2010.

[6] Stephen A. Cook. Can computers routinely discover mathematical proofs? *Proceedings of the American Philosophical Society*, 128(1):40–43, 1984.

[7] Cride5. Move count metrics for big cubes - standards and preferences. Speed Solving Forum, August 2010. URL: `https://www.speedsolving.com/forum/showthread.php?23546-Move-count-metrics-for-big-cubes-standards-and-preferences`.

[8] P. Damaschke. The hamiltonian circuit problem for circle graphs in np-complete. *Inf. Process. Lett.*, 32(1):1–2, July 1989.

[9] H. N. de Ridder et al. Problem: Hamiltonian cycle. Information System on Graph Classes and their Inclusions (ISGCI). URL: `http://www.graphclasses.org/classes/problem_Hamiltonian_cycle.html`.

[10] Erik D. Demaine. Lecture 8: Hamiltonicity. In *MIT 6.890: Algorithmic Lower Bounds*. 2014. `http://courses.csail.mit.edu/6.890/fall14/lectures/L08.html`.

[11] Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Anna Lubiw, and Andrew Winslow. Algorithms for solving Rubik's Cubes. In *Proceedings of the 19th European*

Conference on Algorithms, ESA'11, pages 689–700, Berlin, Heidelberg, 2011. Springer-Verlag.

[12] Michal Forišek. Computational complexity of two-dimensional platform games. In *Proceedings of the 5th International Conference on Fun with Algorithms*, pages 214–227, Ischia, Italy, June 2010.

[13] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, November 1982.

[14] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, Yorktown Heights, New York, March 1972.

[15] Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A survey of NP-complete puzzles. *ICGA Journal*, 31:13–34, 2008.

[16] László Lovász. The matroid matching problem. *Algebraic methods in graph theory*, 1:495–517, 1978.

[17] Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(1):291 – 298, 1996.

[18] Christos H. Papadimitriou and Umesh V. Vazirani. On two geometric problems related to the travelling salesman problem. *Journal of Algorithms*, 5(2):231–246, 1984.

[19] J. Plesník. The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Information Processing Letters*, 8(4):199–201, April 1979.

[20] Daniel Ratner and Manfred Warmuth. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, July 1990.

[21] Shuichi Ueno, Yoji Kajitani, and Shin'ya Gotoh. On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. *Discrete Mathematics*, 72(1):355 – 360, 1988.

[22] Christopher Umans and William Lenhart. Hamiltonian cycles in solid grid graphs. In *Proceedings of the 38th Annual IEEE Conference on Foundations of Computer Science*, pages 496–505, Miami, Florida, October 1997.

[23] Jeffε (`https://cstheory.stackexchange.com/users/111/jeff\%CE\%B5`). Is optimally solving the n×n×n Rubik's Cube NP-hard? Theoretical Computer Science Stack Exchange. URL: `https://cstheory.stackexchange.com/q/783(version:2010-10-23)`.

[24] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935.

[25] Speed Solving Wiki. Metric, May 2010. URL: `https://www.speedsolving.com/wiki/index.php/Metric`.

[26] Takayuki Yato. On the NP-completeness of the Slither Link puzzle. *IPSJ SiG Notes*, AL-74:25–32, 2000.