The Vulcan game of Kal-toh: Finding or making triconnected planar subgraphs

by

Terry David Anderson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the game of *Kal-toh* depicted in the television series *Star Trek: Voyager*, players attempt to create polyhedra by adding to a jumbled collection of metal rods. Inspired by this fictional game, we formulate graph-theoretical questions about polyhedral (triconnected and planar) subgraphs in an on-line environment. The problem of determining the existence of a polyhedral subgraph within a graph G is shown to be NP-hard, and we also give some non-trivial upper bounds for the problem of determining the minimum number of edge additions necessary to guarantee the existence of a polyhedral subgraph in G. A two-player formulation of Kal-toh is also explored, in which the first player to form a target subgraph is declared the winner. We show a polynomial-time solution for simple cases of this game but conjecture that the general problem is NP-hard.

Acknowledgements

I am truly grateful for the unending support, encouragement, and patience of my supervisor, Therese Biedl. I would also like to thank my thesis readers, Craig Kaplan and Alex López-Ortiz, for their feedback and suggestions.

Dedication

To my family.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 The Game of Kal-toh

Kal-toh is a fictional game from the television series *Star Trek: Voyager*. It was first introduced by the character Tuvok in the episode entitled "Alter Ego" from the show's third season. Tuvok belongs to an alien race known as the Vulcans, known for their superior intelligence, mastery of logic, and highly analytical minds. The cultural impact of Kal-toh among the Vulcans can be compared to the Human game of chess; achieving *Grandmaster* status in either game requires intellect, dedication, and a lifetime of study. (However, this comparison is viewed as somewhat insulting to Tuvok, who remarked that "Kal-toh is to chess as chess is to tic-tac-toe.")

Unfortunately, very few details of the game are explained to the viewer, so the following description is largely based on personal interpretation. The game of Kal-toh is either played alone or between two players and uses small metal rods, which appear to be connected to one another at their endpoints. Initially the game appears as a seemingly random structure of interconnected rods in three dimensions. One "move" consists of removing a rod and placing it elsewhere in the structure, with the ultimate goal of forming a polyhedral structure. One episode depicts a polyhedron being formed by using every rod in the structure, whereas another episode depicts a player forming a polyhedron using only a subset of the rods.

This thesis considers a collection of interesting problems in algorithmic graph theory and game theory which are inspired by the game of Kal-toh. After formulating the problems from Kal-toh, our main goal is to provide solutions wherever possible, or at least describe possible approaches and suggest open problems. We give a more detailed introduction to the graph problems inspired by the one-player and two-player versions of Kal-toh in Section 1.1.1.

Following the brief review of graph theory definitions in Section 1.2, we give an overview of related work in Section 1.3. The one-player and two-player variants of Kal-toh are studied

in Chapters 2 and 3, respectively. Finally, Chapter 4 gives a summary of our results and provides suggestions for future work.

1.1.1 Our Interpretation of Kal-toh

Despite a lack of details given about Kal-toh, it seems clear that the the objective of Kal-toh is to place small metal rods at certain locations in order to create a polyhedron in three dimensions. From the limited gameplay one can see in the *Star Trek* episodes, it appears that a rod is removed and then re-added elsewhere within the game structure so that it somehow "fits" among the others, possibly subject to gravity or a magnetic property of the rods.

For simplicity, we ignore any physical constraints imposed by the length or weight of the metal rods and will assume that any rod added to the configuration must be placed exactly between the endpoints of two existing rods. This allows us to use a graph to model any configuration of the game, using edges to represent rods and vertices to represent the connection of two rods at their endpoints. This model also seems appropriate given Tuvok's comment that Kal-toh "is not about striving for balance, but about finding the seeds of order even in the midst of profound chaos." We will also make the assumption that a *new* edge is added to the graph on each move, instead of first being removed from elsewhere in the graph. This assumption is made in order to make a single move as simple as possible.

We can then view a configuration of the game as a simple, non-geometric graph G. Steinitz's Theorem states that a graph G is the edge graph of a polyhedron if and only if G is a simple graph which is triconnected and planar. Using this theorem, we see that finding polyhedra in three dimensions is equivalent to finding triconnected planar graphs, and can be formalized as follows.

Let G be a graph having a fixed set of n vertices and an edge set which is possibly empty. The graph G is modified incrementally by the addition of edges (multiple edges between two vertices are not allowed). We are interested in the existence of any subgraph H of G such that H is both triconnected and planar; it is not necessary for G itself to satisfy either of these conditions. With this formulation in mind, we can ask the following questions:

- Does there exist a subgraph H of G such that H is both triconnected and planar?
- If at least one such H exists:
  - Which subgraph(s) has/have the largest number of vertices?
  - Which subgraph(s) is/are maximally planar (i.e., the addition of any one edge would violate planarity)?
  - Which subgraph(s) is/are minimally triconnected (i.e., the removal of any one edge would violate triconnectivity)?

- If no such H exists:
  - Can a triconnected planar subgraph H of G be created by adding a single edge to G?
  - Can a triconnected planar subgraph H of G be created by adding $l > 1$ edges to G?
  - What is the minimum number of edges that must be added to G such that there exists a triconnected planar subgraph H of G?
  - What is the maximum number of edges that can be added to G without creating a triconnected planar subgraph H of G? In other words, what number of edges guarantees the existence of such an H?

To study the answers to some of these questions, we formulate the one-player version of Kal-toh as the following two-parameter decision problem:

KAL-TOH$_{l,\geq k}(G)$ Let G be a graph on a fixed set of $n \geq 4$ vertices. Does there exist a triconnected planar subgraph H of G on at least $4 \leq k \leq n$ vertices after the addition of at most l edges?

The problem KAL-TOH$_{l,=k}(G)$ is used when we wish to specify the *exact* number of vertices required for the triconnected planar subgraph. We will generally assume G to be a fixed graph and hence use the notation KAL-TOH$_{l,\geq k}$ to mean the same thing as KAL-TOH$_{l,\geq k}(G)$.

Our interpretation of the one-player game appears to be a mixture of problems related to incremental planarity testing and triconnectiviy augmentation, both of which are described in Section 1.3. But to the best of our knowledge, the KAL-TOH$_{l,\geq k}(G)$ problem has not previously been studied.

Chapter 2 examines various instances of this problem in further detail. Section 2.1 considers KAL-TOH$_{0,\geq 4}$, which asks if G contains a triconnected planar subgraph H. We show that it is NP-complete to answer this question. Section 2.2 considers the case $l > 0$ in which the addition of edges is allowed. In addition to some results on simple instances of the game, we show that the addition of at most $k + 1$ edges to a connected graph G always suffices to form a triconnected subgraph H with k vertices. Our result is proved by adding edges to a spanning tree so that the resulting graph has a so-called Schnyder wood. We then prove that any graph with a Schnyder wood and a triangle between the roots is triconnected, a result of possible independent interest.

Chapter 3 considers the two-player variant of Kal-toh in which players add edges to the graph in an alternating manner. To reduce the complexity, instead of attempting to create *any* triconnected planar subgraph of G, we model this as a game where the first player to create a prescribed graph H as a subgraph of G is declared the winner. We call this game

KAL-TOH-2P$_n(H)$ (with G usually being the graph on n isolated vertices) which informally asks which player has a strategy allowing them to be the first to create H as a subgraph of G. A general approach to the analysis of this problem is given in Section 3.1, along with results for simple cases. Section 3.2 looks at the cases where H is a four-cycle or a diamond in greater depth, the analysis of which becomes surprisingly difficult. This leads us to conjecture that there is no polynomial-time solution to KAL-TOH-2P$_n(H)$ for all graphs H.

1.2 Background

Here we give a brief overview of some basic definitions and results in graph theory that are relevant to this thesis. Additional details on these topics can be found in any textbook on graph theory or graph algorithms, such as [5] or [9].

1.2.1 Graphs

A *graph* $G = (V, E)$ consists of a set of *vertices* V and a set of *edges* E such that each edge $e = (v_1, v_2) \in E$ has two vertices $v_1 \in V$ and $v_2 \in V$ as its *endpoints*. In this case we say that the vertices v_1 and v_2 are *adjacent* (or *neighbours*). We will assume that G is *simple* unless otherwise indicated, which means that the endpoints of each edge are distinct (no *self-loops*) and any two vertices are the endpoints of at most one edge (no *multiple edges*). The edge e is *incident* to vertex v if v is an endpoint of e, and the *degree* of a vertex v is the number of edges incident with v, denoted as $\deg(v)$. The *Handshaking Lemma* states that $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.

Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are called *isomorphic* if $|V_G| = |V_H|$ and there exists a bijective mapping $v \leftrightarrow v'$ between vertices $v \in V_G$ and $v' \in V_H$ with the property that $(v_1, v_2) \in E_G$ if and only if $(v_1', v_2') \in E_H$.

Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be graphs. H is said to be a *subgraph* of G if $V_H \subseteq V_G$ and $E_H \subseteq E_G$ and a subgraph H of G is called a *proper subgraph* if $H \neq G$. Let V' be a subset of vertices of the graph G. Then the *induced subgraph* $G[V']$ is the graph (V', E') where $E' = \{(v_1, v_2) : v_1 \in V', v_2 \in V', (v_1, v_2) \in E_G\}$. Every induced subgraph of G is also a subgraph of G, but not every subgraph of G is an induced subgraph of G. When the term *subgraph* is used, it does not necessarily refer to an induced subgraph. In this thesis, we use the phrase G *contains* H to mean that H is a subgraph of G.

A *path* P is a list of vertices $(v_1, v_2, v_3, \ldots, v_k)$ such that v_i and v_{i+1} are adjacent for all $1 \leq i \leq k-1$ and $v_i \neq v_j$ if $i \neq j$, for all $1 \leq i, j \leq k$. P is called a *k-path* if it contains $k+1$ vertices (equivalently, k edges) and is said to have *length* k. If a path P is a subgraph of G such that each vertex in V_G appears exactly once in P, then P is called a *Hamiltonian path*.

A *cycle* C is a path $(v_1, v_2, v_3, \ldots, v_k)$ except that $v_1 = v_k$. A *k-cycle* C_k is a cycle on k vertices (equivalently, k edges) and is said to have length k. The 3-cycle is often called a

triangle. If a cycle C is a subgraph of G such that each vertex in V_G appears exactly once in C, then C is called a *Hamiltonian cycle*. The *girth* of a graph G is the length of the shortest cycle appearing as a subgraph of G.

If each vertex in G has degree k, it is said to be *k-regular*. A *complete* graph is a special type of regular graph having an edge between any pair of two vertices. The complete graph on n vertices is denoted K_n. Note that a complete graph has $\binom{n}{2}$ edges, and this gives an upper bound on the number of edges present in any graph G.

A graph $G = (V, E)$ is called *bipartite* if there exists a way to partition the vertices of G into two disjoint sets $V = A \dot{\cup} B$ such that $v_1 \in A$ and $v_2 \in B$ for every edge $(v_1, v_2) \in E$. The *complete bipartite graph* contains an edge between each pair of vertices $v_1 \in A$ and $v_2 \in B$, denoted as $K_{a,b}$ where $a = |A|$ and $b = |B|$. This graph has $a \cdot b$ edges.

1.2.2 Connectivity

A graph G is *connected* if there exists a path between any two of its vertices. If there does not exist a path between two vertices x and y, these vertices are said to be in different *components* of the graph G. If the removal of a single vertex v disconnects the graph, v is called an *articulation point*. If the removal of a single edge e disconnects the graph, e is called a *bridge*.

A graph G is *k-connected* (or *k-vertex-connected*) if it remains connected after the removal of any set of $k-1$ vertices, and it is *k-edge-connected* if it remains connected after the removal of any set of $k-1$ edges. *Menger's theorem* states that G is k-vertex-connected (k-edge-connected) if and only if there exist k pairwise vertex-disjoint (edge-disjoint) paths between any two vertices in G. A 2-connected graph is also called *biconnected*, and a 3-connected graph is also called *triconnected*. All triconnected graphs must contain at least four vertices.

1.2.3 Trees

A connected graph with no cycles is called a *tree*. A tree on n vertices always has $n-1$ edges, and there exists a unique path between any two vertices. The removal of any one edge will disconnect the tree (i.e. every edge is a bridge), and the addition of any one edge will create a cycle. If a tree $T = (V_T, E_T)$ is a subgraph of a graph $G = (V_G, E_G)$, then T is called a *spanning tree* of G if $V_T = V_G$. A vertex v in a tree is called a *leaf* if $\deg(v) = 1$. A *star* is a tree on n vertices with $n-1$ leaves. The general name for any graph with no cycles is a *forest*; each connected component of a forest is a tree.

In a *rooted tree* T, one vertex r is assigned to be the *root* vertex. If the length of the path between r and another vertex v is l, then the vertex v appears at *level* $l+1$ in T; the root vertex appears at level 1. The number of levels in a tree is called its *height*. Let u and v be two adjacent vertices in T such that u appears at level l and v appears at level $l+1$. We say that v is a *child* of u and that u is the *parent* of v. Two vertices with a common parent

are called *siblings*. Note that all vertices have exactly one parent (except for the root), but may have any number of children.

Let u and v be two vertices in T such that the level of u is no more than the level of v. If the root vertex r does not appear on the path between u and v, then v is called a *descendant* of u and u is called an *ancestor* of v. By definition, any vertex is also its own ancestor and its own descendant.

1.2.4 Planarity

A graph G is called *planar* if it can be drawn on the surface of a sphere without any two of its edges crossing. We will instead talk about graphs in $\mathbb{R}^2$, as any graph on the surface of a sphere can be expressed on the plane via the stereographic projection. A specific realization of a graph on the plane is called a *drawing* of the graph. This divides the plane into disjoint regions called *faces*, and there is always exactly one face containing infinity called the *outer-face*.

If G is a simple connected planar graph on n vertices, m edges, and f faces, then *Euler's formula* states that $n - m + f = 2$. A corollary to this formula is the inequality $m \leq 3n - 6$. In an *edge-subdivision* of a graph G, additional vertices may be placed along any edge of G. *Kuratowski's Theorem* states that G is a planar graph if and only if it contains an edge-subdivision of K_5 or $K_{3,3}$.

1.2.5 Polyhedral Graphs

A *(convex) polyhedron* is a finite region defined by the intersection of half-spaces in $\mathbb{R}^3$ (we will not consider non-convex polyhedra in this thesis). For a polyhedron P, the corresponding graph $G = (V, E)$ can be constructed by defining a vertex $v \in V$ for each extreme point of P and an edge $e \in E$ for each extreme line segment of P. G is called the *polyhedral graph* of P. *Steinitz's Theorem* states that the polyhedral graph G corresponding to a convex polyhedron P is planar and triconnected and, furthermore, each triconnected planar graph G is the polyhedral graph of some convex polyhedron P. For instance, the graph K_4 corresponds to the tetrahedron.

Every triconnected graph must have minimum degree at least three. By the Handshaking Lemma, $\frac{3n}{2}$ is the smallest number of edges in a triconnected graph. Because a polyhedral graph is also planar, we apply Euler's formula to see that $\frac{3n}{2} \leq m \leq 3n-6$ for any polyhedral graph G on n vertices and m edges. *Whitney's theorem* states that a polyhedral graph has exactly two *planar embeddings*, i.e., an ordering of the edges around each vertex [8].

1.3 Related Work

The single-player version of Kal-toh is related to the problems of planarity testing and k-connectivity testing, both in a static environment and in an on-line environment where edge additions are allowed. Its two-player variant also closely resembles problems of subgraph isomorphism.

Kal-toh is an interesting problem to consider due to the differences in the properties of planarity and k-connectivity. If G is a triconnected planar graph, then the property of planarity is maintained for any subgraph of G, but the same cannot be said for triconnectivity. On the other hand, the property of triconnectivity is maintained for all supergraphs of G, but it is not guaranteed that these graphs will also be planar.

In this section we provide a listing of some known results in each of these areas and finish by giving an overview of results about planar c-connected subgraphs, some of which are shown in this thesis.

1.3.1 Planarity

Recall that a winning game of Kal-toh (for a single player) is a graph G containing a triconnected planar subgraph H. Either a graph already contains such a subgraph, or it can be obtained by the addition of edges. We are therefore interested in related results about testing the planarity of a given graph and about how edge additions may violate planarity. Note that our focus is slightly different because a game of Kal-toh does not require the graph G to be planar.

The problem of *planarity testing* asks if there exists a planar embedding of a graph G. After the original formulation of this problem was given by Goldstein [16], Hopcroft and Tarjan showed a $O(n)$-time algorithm to test the planarity of a graph G [23]. Their approach first divides the graph into its biconnected components and then tests planarity in a recursive manner through the identification of cycles in each component. Unfortunately, this algorithm only answers the decision problem and does not give any further information about the graph. Mehlhorn et al. [33] modify the algorithm in order to output a combinatorial embedding if G is planar or the location of a K_5 or $K_{3,3}$ edge-subdivision otherwise. It should be noted that there are many other approaches to planarity testing; see [21] for an overview.

The *incremental planarity testing* problem is considered in an online environment where the addition and removal of vertices and edges are allowed. In the context of Kal-toh, the only operation we need to consider is the addition of edges. If G is biconnected, a data structure requiring $O(n)$ space is described by Di Battista et al. that supports edge additions in $O(\log n)$ amortized time and answers the query "can the edge (v_1, v_2) be added to G while maintaining planarity?" in $O(\log n)$ time [7]. If the biconnectivity requirement on G is dropped, Galil et al. describe a data structure supporting edge insertions in $O(n^{\frac{2}{3}})$ amortized time and planarity testing in $O(n^{\frac{2}{3}})$ time [14].

1.3.2 k-Connectivity and k-Connected Components

In addition to being planar, the target subgraph H in a game of Kal-toh must also be triconnected. Recognizing if such an H exists is related to the problem of triconnectivity testing. However, an important difference is that we do not require G to be triconnected.

Some of the results discussed in this thesis will require that a graph G is biconnected as a precondition. A data structure known as a BC-tree can be used to represent the structure of the vertex pairs whose removal disconnects a graph G (i.e, *cutvertices*) and the corresponding biconnected components. The procedure for finding the biconnected components using a depth-first search are given by Tarjan [38] and additional details of BC-trees are given by Kant [28]. A graph G is biconnected if and only if no cutvertices appear in its BC-tree. Since the tree can be constructed in linear time [5], it is therefore possible to test biconnectivity in linear time.

Testing the triconnectivity of a graph G follows a similar idea. An SPQR-tree is a data structure that stores information about the triconnected components of G. Hopcroft and Tarjan give an $O(m + n)$-time algorithm to decompose a graph into its triconnected components [22], further discussed in [19]. See [8] for further details about SPQR-trees, or [25] for information about the related 3-block tree data structure. The SPQR-tree of a biconnected graph can be constructed in linear time [8, 19].

1.3.3 Connectivity Augmentation

In the game of Kal-toh where we allow the addition of edges, we are interested in *connectivity augmentation* problems, which ask if G is k-connected after the addition of one or more edges. Similar variants exist when the addition of vertices are allowed, but these are not applicable to the game of Kal-toh. As already mentioned, our problem differs from existing results because we want to know the connectivity of a subgraph and not necessarily the entire graph G.

Hsu and Ramachandran give a linear-time algorithm to solve the *biconnectivity augmentation* problem, which finds a minimum set of edge additions to biconnect G [26]. The same authors also give a linear-time algorithm to solve the *triconnectivity augmentation* problem, defined similarly [25]. This result is an improvement on the previous bound of $O(n(n+m)^2)$ by Watanabe and Nakamura [39].

However, these problems become more complicated when the planarity of G must be maintained. Any planar graph can be made triconnected by triangulating, but the difficult part is minimizing the number of edges needed to do so. The *planar biconnectivity augmentation* problem (PBA) asks for a minimum set of edges that will biconnect a graph while maintaining planarity; the *planar triconnectivity augmentation* problem (PTA) is defined similarly. Both of these problems are NP-hard in general [29]. See [34] and [27] for an alternate formulation of PBA and PTA as integer programming problems.

An approximation algorithm for PBA with performance ratio $\frac{3}{2}$ and an approximation algorithm for PTA with performance ratio $\frac{5}{4}$ is given in [29]. Unfortunately, the approximation for PBA as it appears in this paper is incorrect; a revised version with performance ratio $\frac{5}{3}$ appears in [20]. Furthermore, [20] also shows the NP-hardness of two weaker versions of PBA: the case where all cutvertices belong to the same biconnected component and the case where the SPQR-tree of this biconnected component has diameter at most two. Details of *four-connectivity augmentation* problems can be found in [24].

All of the results mentioned so far in this section apply to the *entire* graph. A related problem requiring only (1-)connectivity that instead applies to a subgraph was studied by Gutwenger et al. [18]: Consider a planar graph $G = (V, E)$, a subset W of vertices, and the subgraph E_W induced by W. A *subgraph induced planar connectivity augmentation* for W is a set of F additional edges with end vertices in W such that $G' = (V, E \cup F)$ remains planar and the subgraph induced by W is connected. Gutwenger et al. give a linear-time algorithm (based on SPQR-trees) which tests the existence of such an augmentation and, if one exists, returns an augmentation of minimum cardinality [18]. This does not direcly apply to the problems we will be studying in this thesis, which require triconnectivity and do not require the subgraph to be induced; they also do not fix the vertex set on which the subgraph should reside.

1.3.4 Finding Subgraphs

Here we list some problems which consider the existence of certain subgraphs H within a graph G; the two-player version of Kal-toh studied in Chapter 3 is a special case where H is both triconnected and planar.

Given graphs G and H, the general SubgraphIsomorphism problem asks whether H appears as a subgraph of G. Even if H is restricted to be a planar graph, this problem is known to be NP-complete [15]. However, the problem PlanarSubgraphIsomorphism where both G and H are restricted to be planar graphs, is fixed-parameter tractable in k, the number of vertices of H [11, 12, 10].

In the single-player version of Kal-toh studied in Chapter 2, we are not given a specific H to find as a subgraph of G. The goal is instead to check the presence of a triconnected planar subgraph H of G such that H has at least $k \geq 4$ vertices. The related MaximumPlanarSubgraph problem asks a similar question, but only requires H to be planar (not necessarily triconnected), and returns a planar subgraph H of G containing a maximum number of edges (instead of vertices). MaximumPlanarSubgraph is NP-complete [31] and has approximation algorithm with a performance ratio of $\frac{2}{5}$ [6]. The problem of finding a maximal *induced* planar subgraph is also NP-hard [40, 39, 41].

1.3.5 c-connected Planar Subgraphs

The problem KAL-TOH$_{0,\geq k}(G)$ is a special case of the general question "Does the graph G contain a planar c-connected subgraph H?", possibly with a restriction on the minimum number of vertices k in H.

We briefly list here results for smaller values of c. For $c = 1$, this question is solvable in polynomial time with or without a restriction on the minimum value of k by finding a spanning tree of G. For $c = 2$, the problem is easy if $k = 0$, i.e., if we want to find any 2-connected planar subgraph. The answer is YES if and only if G contains any cycle, i.e., if and only if G is not a tree.

For $c = 2$ and $k > 0$, we suspect that it is NP-hard, because it somewhat resembles the HAMILTONIANCYCLE problem ("Does G contain a cycle of length at least k?"). However, we only require that H is biconnected and planar, not that it is a cycle, so proving NP-hardness remains open.

When moving from $c = 2$ to $c = 3$, the problem makes a jump from having a polynomial-time solution to being NP-hard. Theorem 1 shows that it is NP-hard to determine if a graph G contains a planar triconnected subgraph.

Chapter 2

Kal-toh for One Player

For a graph G on a fixed set of $n \geq 4$ vertices, recall that KAL-TOH$_{l,\geq k}$(G) is the question, "does there exist a triconnected planar subgraph H of G on at least $4 \leq k \leq n$ vertices after the addition of at most l edges?" The problem KAL-TOH$_{l,=k}$(G) is used when we wish to specify the *exact* number of vertices required for the triconnected planar subgraph.

In this chapter, we look at specific parameterizations of KAL-TOH$_{l,\geq k}$. Section 2.1 begins by considering KAL-TOH$_{0,\geq 4}$: without requiring the addition of edges, does a graph G contain *any* triconnected planar subgraph? We show this problem to be NP-complete and also give a similar proof for the NP-completeness of KAL-TOH$_{0,\geq \frac{3n}{4}}$. Section 2.2 then considers instances of KAL-TOH$_{l,=k}$ in which edges are added to the graph. In addition to some specific results for small values of l, we prove that the answer to KAL-TOH$_{k+1,=k}$ is YES when G is connected.

2.1 Recognizing a Winning Graph

Our reduction used in proving the NP-hardness of KAL-TOH$_{0,\geq 4}$ will be from the NONCROSSINGCYCLE problem in an orthogonal geometric graph G, defined below.

Definition 1 (Orthogonal Geometric Graph, Crossing)**.** *An **orthogonal geometric graph** is a graph drawn in the plane such that each edge is represented by a path of contiguous axis-parallel line segments such that no two line segments have infinitely many points of intersection, and no segment intersects a non-incident vertex. A **crossing** is a point that belongs to the interior of two segments of edges.*

The NONCROSSINGCYCLE problem, defined below, was shown to be NP-complete in [30].

Definition 2 (NoncrossingCycle Problem)**.** *For an orthogonal geometric graph G, the* NONCROSSINGCYCLE *problem asks if there exists a cycle in G that contains no crossing.*

Let G be an orthogonal geometric graph. Preprocess the graph G by placing a dummy vertex at each bend, which is the common point of any two consecutive segments belonging to the same edge. Furthermore, if two line segments l_1 and l_2 both cross a line segment l_3, place a vertex along l_3 between these crossings. This preprocessing of G guarantees that each crossing is surrounded by four vertices. Note that this does not change our problem; adding the dummy vertices will neither add nor remove a non-crossing cycle from G. Figure 2.1 shows an example of an orthogonal geometric graph containing a non-crossing cycle.

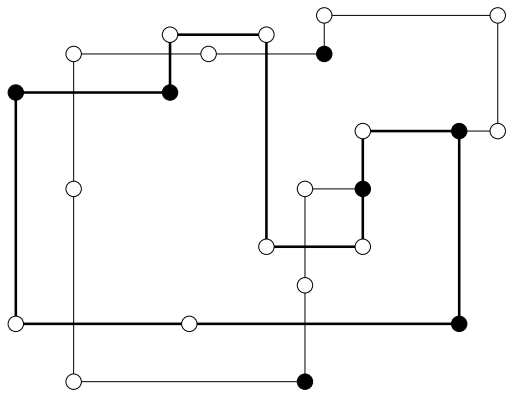

Figure 2.1: An orthogonal geometric graph (black vertices) with a non-crossing cycle (in bold). White vertices are added in the pre-processing step.

From G, we construct the graph H that is the instance for KAL-TOH$_{0,\geq 4}$ by replacing each vertex and crossing in G by a *spine* in H, described in Definition 3.

Definition 3 (Vertex-spine, Crossing-spine, Spine)**.** *A vertex $v \in G$ is replaced with a two-path $(s_1(v), s_2(v), s_3(v)) \in H$, called a **vertex-spine**. A crossing $c \in G$ is replaced with a two-path $(s_1(c), s_2(c), s_3(c)) \in H$, called a **crossing-spine**. A **spine** is either a vertex-spine or a crossing-spine.*

An edge $e \in G$ by our preprocessing consists of a horizontal or vertical segment with at most one crossing and with a vertex at both ends. This edge is replaced in H as follows. If there is no crossing along e, it is represented by a *vertex-segment gadget*, described in Definition 4. If there is a crossing along e, it is represented by two *crossing-segment gadgets*, described in Definition 5. A *segment gadget* is either a vertex-segment gadget or a crossing-segment gadget.

Definition 4 (Vertex-segment gadget)**.** *Let (v, w) be an edge in G that does not contain a crossing. In H, v and w are replaced with spines (defined previously) and the edge (v, w) is replaced with the six-cycle $(x_3, x_2, x_1, y_1, y_2, y_3)$ such that $s_i(v)$ is connected to x_i for $i = 1, 2, 3$ and $s_i(w)$ is connected to y_i for $i = 1, 2, 3$. See Figure 2.2.*

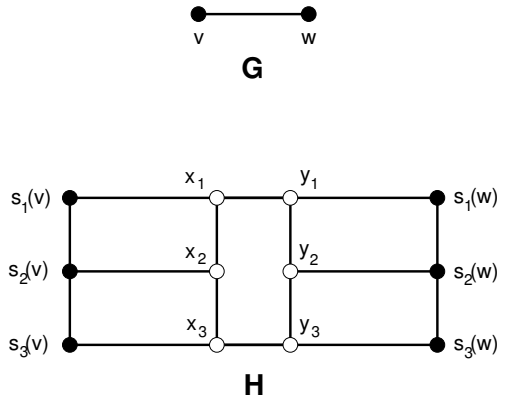

Figure 2.2: For an edge $(v, w) \in G$ without a crossing, vertices v and w are replaced with vertex-spines in H, and these spines are connected to a common vertex-segment gadget.

Definition 5 (Crossing-segment gadget, Opposite crossing-segment gadgets)**.** *Let (v, w) be an edge in G that contains a crossing c. In H, v, w, and c are replaced with spines (defined previously). We add the seven-cycle $D = (x_3, x_2, x_1, x_0, y_1, y_2, y_3)$ to H (called a **crossing-segment gadget**) such that $s_i(v)$ is connected to x_i for $i = 1, 2, 3$ and $s_i(c)$ is connected to y_i for $i = 1, 2, 3$. We also add the the seven-cycle $D' = (x'_3, x'_2, x'_1, x'_0, y'_1, y'_2, y'_3)$ connecting the spines for c and w in a similar manner. Finally, we add an edge between x_0 and x'_0. The crossing-segment gadgets D and D' are called **opposite**. See Figure 2.3.*

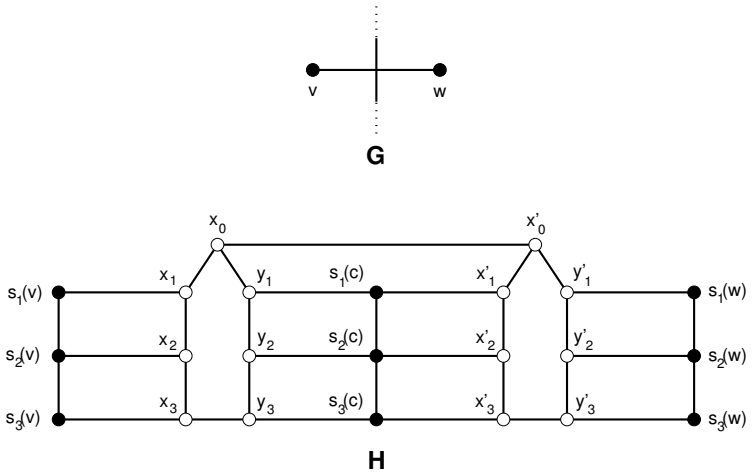

Figure 2.3: For an edge $(v, w) \in G$ with a crossing c, vertex v, vertex w, and c are replaced with spines in H, and these spines are connected using two opposite crossing-segment gadgets.

A more detailed example is shown in Figure 2.4.

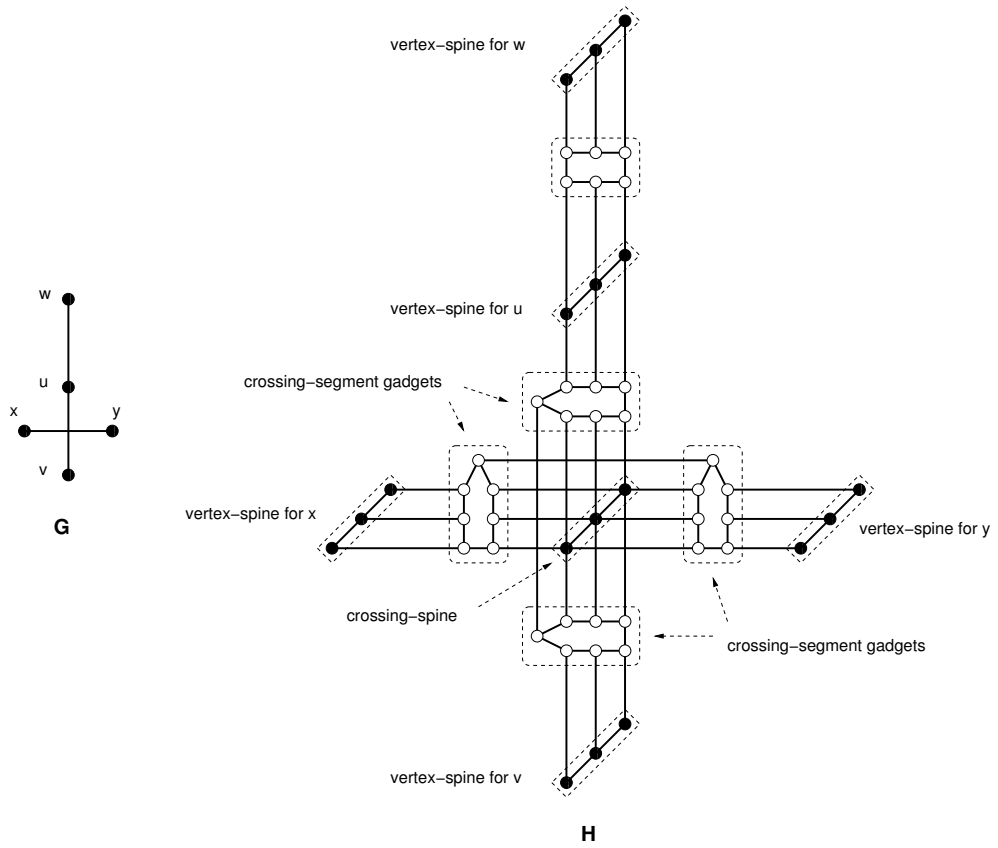

Figure 2.4: The graph H is constructed from the graph G using spines and segment gadgets.

This ends the description of H. We will now show that G has a non-crossing cycle if and only if H has a triconnected planar subgraph.

First assume we are given a non-crossing cycle C of G. We can construct a corresponding triconnected planar subgraph H' of H as follows. Consider any edge (v_1, v_2) in C, and let S_1 and S_2 be the vertex-spines of v_1 and v_2, respectively, in H. Then in H' we include S_1 and S_2. If (v_1, v_2) has no crossing, then we include its vertex-segment gadget. If it does, we include its crossing-spine and and both crossing-segment gadgets. It is easy to see that the subgraph H' is triconnected. H' is also planar because this graph can be drawn without crossings, as shown in Figure 2.5.

To show the other direction, let H' be any triconnected planar subgraph of H. We think of H' as being obtained from H by removing enough edges to achieve planarity while at the same time maintaining triconnectivity. The following four lemmas give conditions implied by the triconnectivity and planarity of H'.

Lemma 1. *For any segment gadget in H, either all or none of the edges incident to its vertices are in H'.*

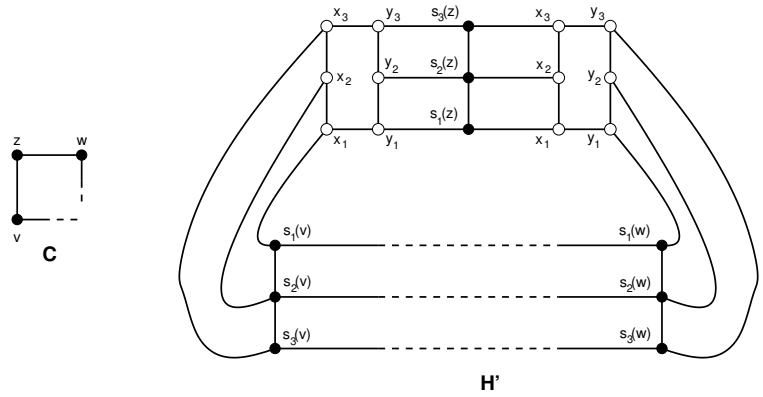

Figure 2.5: A non-crossing cycle C in G corresponds to the triconnected subgraph H' of H. As illustrated here, there exists a drawing of H' without crossings.

Proof. Consider a vertex-segment gadget in H'. Each of the vertices have degree exactly three, so removing an incident edge from this gadget would violate the triconnectivity of H' by changing the degree of a vertex to two. In this case, every other edge belonging to the gadget would need to be removed in order to guarantee no vertices of degree two. Therefore either all or none of the edges must be present in a vertex-segment gadget in H'. A similar argument holds for a crossing-segment gadget. □

Lemma 2. *Any spine in H is connected to at most two non-empty segment gadgets in H'.*

Proof. Consider a spine in H' made up of vertices s_1, s_2, and s_3 that is connected to at least three segment gadgets. By Lemma 1, each of these segment gadgets contain all of their edges, and also all edges to vertices s_1, s_2, and s_3. Let a, b, and c be the vertices in the three segment gadgets that are adjacent to s_2. Then we have that the two sets $\{s_1, s_2, s_3\}$ and $\{a, b, c\}$ give an edge-subdivision of $K_{3,3}$, violating planarity of H'. See Figure 2.6. □

Lemma 3. *Any spine in H is connected to either 0 or 2 non-empty segment gadgets in H'.*

Proof. Using Lemma 2, we only need to argue that it is impossible for a spine to have exactly one adjacent segment gadget that is non-empty in H'. For a contradiction, assume that the spine $S = (s_1, s_2, s_3)$ has exactly one non-empty adjacent segment gadget D. By Lemma 1, the vertices of S are adjacent *only* to the vertices of D. But this means that s_1 and s_3 have degree exactly two, which is a contradiction to the triconnectivity of H'. □

Lemma 4. *If a crossing-spine in H is connected to two crossing-segment gadgets R_1 and R_2 that are both non-empty in H', then R_1 and R_2 are opposite.*

Proof. Let S be a crossing-spine connected to exactly two non-empty crossing-segment gadgets in H' which are *not* opposite. Consider one of these crossing-segment gadgets. Because

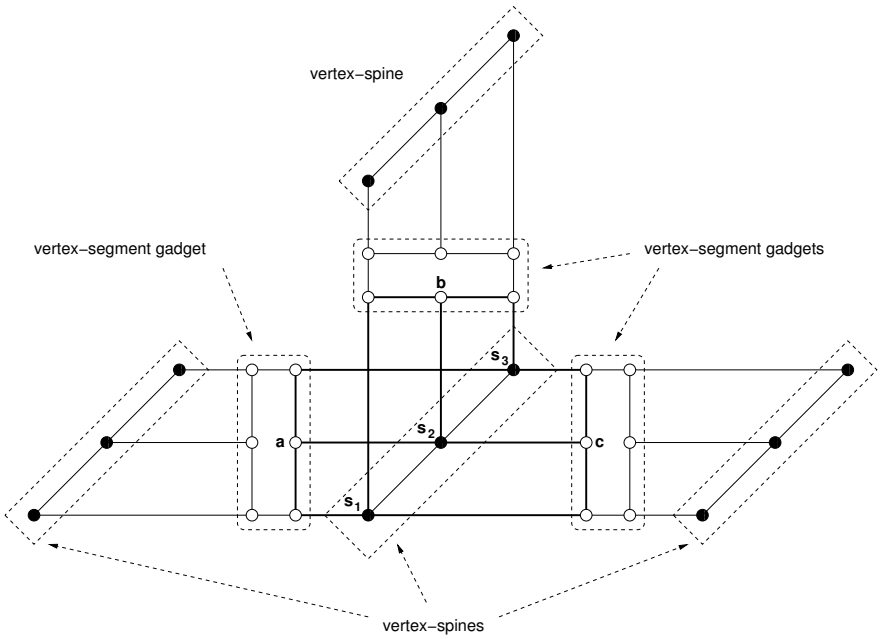

Figure 2.6: An edge-subdivision of $K_{3,3}$ exists (indicated using bolded edges between $\{s_1, s_2, s_3\}$ and $\{a, b, c\}$) when a vertex-spine is adjacent to at least three other vertex-spines.

there is no opposite crossing-segment gadget in H', vertex x_0 of this segment gadget is missing an edge, which contradicts Lemma 1. □

So given a triconnected planar subgraph H' of H, we define C to be a subset of the edges of G as follows. Consider any edge (v_1, v_2) in the graph G. Then the edge (v_1, v_2) is in the set C if and only if and only if the segments replacing (v_1, v_2) in H belong to H' (they must be in H' entirely or not at all by Lemmas 1 and 4).

We first note that the set C is non-empty. For otherwise, H' would be a subforest of paths on three vertices by Lemma 1 because no two spines would be connected by a common segment gadget. But this is a contradiction because H' must be triconnected. Therefore by Lemma 3, the vertices in C all have degree two and must form a cycle in G because H' is triconnected. Furthermore, Lemma 4 guarantees that C is a non-crossing cycle.

A solution to our problem can be verified in polynomial time by performing planarity and triconnectivity tests on the graph H', so it is in NP. Furthermore, our reduction requires polynomial time.

Theorem 1. *It is* NP*-complete to determine if a graph G contains a triconnected planar subgraph.*

Note that Theorem 1 also holds if we were to replace "subgraph" by "induced subgraph" because in no part of our proof do we ever specifically require that H' be a non-induced

subgraph. Theorem 1 also holds if we were to replace "triconnected" by "having minimum vertex-degree of 3," or if we add the restriction that G is triconnected.

Corollary 1. KAL-TOH$_{0,\geq 4}$ *is* NP*-complete.*

Recall that KAL-TOH$_{0,\geq k}$ asks whether or not a given graph G has a triconnected planar subgraph on at least k vertices. By Theorem 1, KAL-TOH$_{0,\geq k}$ is NP-hard because it is already NP-hard for the special case $k = 4$.

One might wonder whether KAL-TOH$_{0,\geq k}$ becomes easier as the value of k gets larger. For example, if this question were instead asked for *induced* subgraphs, then KAL-TOH$_{0,\geq k}$ for the case $k = n$ is easily answered by running planarity and triconnectivity tests on the input graph G.

With a very similar proof, reducing from HAMILTONIANCYCLE in 3-regular graphs, we can show that KAL-TOH$_{0,\geq k}$ remains NP-complete for $k = \frac{3n}{4}$ in Theorem 2.

Theorem 2. KAL-TOH$_{0,\geq \frac{3n}{4}}$ *is* NP*-complete.*

Proof. Let G be an instance of the HAMILTONIANCYCLE problem for 3-regular graphs. We define H by replacing each vertex in G by a vertex-spine and replacing each edge in G by a vertex-segment gadget.

Given a triconnected planar subgraph H' of H, Lemmas 1, 2, and 3 continue to hold. So we can define the set C using H' as before, which must be a cycle of G.

If G has n vertices and m edges, then it can be verified that H has $\frac{3n}{2}$ edges and $3n+6m = 3n + 6\left(\frac{3n}{2}\right) = 12n$ vertices. We are demanding that $|V(H')| \geq \frac{3}{4}|V(H)| = 9n$ and we know that $|V(H')| = 9|C| \leq 9n$, which implies $|V(H')| = 9n$. So $|C| = n$ and thus C must be a Hamiltonian cycle of G.

For the other direction, assume we are given a Hamiltonian cycle C of G. In a similar manner to our argument for Theorem 1, it can be shown that this Hamiltonian cycle corresponds to a triconnected planar graph H'.

□

2.2 Creating a Winning Graph

We now consider the one-player version of Kal-toh in which edge additions are allowed to create a triconnected planar subgraph H of G.

Section 2.2.1 looks at the problem KAL-TOH$_{l,\geq 4}$, creating an instance of K_4 as a subgraph of G by adding at most l edges. After demonstrating simple polynomial-time algorithms to solve this problem for $l \geq 3$, we observe that our approaches do not work for $l = 2$ or $l = 1$.

We suspect the problem to be NP-complete in general when H can be *any* planar triconnected subgraph, but the problem has a polynomial-time solution when G is planar and H is any fixed graph. To solve this, we must consider the addition of at most l edges, which corresponds to at most $2l$ vertices. For each of the $O(n^{2l})$ possible sets of edge additions, it then takes linear time in $|H|$ to determine if G contains a subgraph isomorphic to H [11].

We then consider the general problem KAL-TOH$_{l,=k}$ for any $k \geq 4$. In Section 2.2.2 we give an upper bound for l by creating a prism as our triconnected planar subgraph H'. In Section 2.2.3 we use a different approach to show a second upper bound of k for l if the graph G is connected. Finally, suggestions for future work appear at the end of the chapter.

2.2.1 Creating K_4

We begin by considering the problem KAL-TOH$_{l,\geq 4}$ on an input graph G in which we aim to create a triconnected planar graph by adding l edges. We will actually create K_4, the complete graph on four vertices, as a subgraph of G. Throughout, we assume that G has at least four vertices, otherwise the answer is always NO. It is clear that we only need to consider values $l \leq 6$ because K_4 can always be constructed as a subgraph of G by adding up to six edges between any four vertices, which immediately implies Theorem 3.

Theorem 3. *The answer to* KAL-TOH$_{6,\geq 4}$ *is always* YES.

A similar argument shows that both KAL-TOH$_{5,\geq 4}$ and KAL-TOH$_{4,\geq 4}$ can be answered in linear time. For $l = 5$, a subgraph of K_4 can be constructed by adding up to five edges between the endpoints of any preexisting edge of G and any two other vertices. For $l = 4$, a subgraph of K_4 can be constructed by adding four edges between the endpoints of any two preexisting edges of G (plus another vertex if the two edges are incident to a common vertex). Finding such vertices and edges can easily be accomplished in linear time.

Theorem 4. KAL-TOH$_{5,\geq 4}$ *and* KAL-TOH$_{4,\geq 4}$ *can be answered in linear time.*

Finally, we argue that KAL-TOH$_{3,\geq 4}$ can also be answered in linear time. By finding a subgraph of G equal to K_4 having exactly three edges removed, as described in Lemma 5, we are able to form a triconnected planar subgraph by adding (at most) three edges to G.

Lemma 5. *If we can create any triconnected planar subgraph H of G by adding at most three edges, then G must have originally contained one of the graphs seen in Figure 2.7.*

Proof. For a contradiction, assume that G did not originally contain any of the graphs seen in Figure 2.7. Then the components of G are single vertices, single edges, and 2-paths. Let H be a triconnected planar graph formed by adding at most three edges to G, and let $G|_{V(H)}$ denote the graph G restricted to the vertices of H (excluding the added edges). The average degree of $G|_{V(H)}$ is at a maximum when all components are 2-paths.

If there are t 2-paths, then the average degree of this graph is at most $\frac{2 \cdot 2t}{3t} = \frac{4}{3}$. Therefore $|E\left(G|_{V(H)}\right)| = \frac{1}{2}\sum_{v \in V(H)} \deg_{G|_{V(H)}}(v) \leq \frac{1}{2} \cdot \frac{4}{3}|V(H)| \leq \frac{2}{3}|V(H)|$.

We have $\sum_{v \in H} \deg_H(v) = 2 \cdot |E(H)|$ by the Handshaking Lemma, and we also have $|E(H)| \geq \frac{3}{2} \cdot |V(H)|$ by the triconnectivity of H. If $a \leq 3$ is the number of edges added to H, then we have $\frac{3}{2}|V(H)| \leq |E(H)| = |E\left(G|_{V(H)}\right)| + a \leq \frac{2}{3}|V(H)| + a$. But $|V(H)| \geq 4$ by triconnectivity, so this implies $a \geq \frac{5}{6}|V(H)| \geq \frac{20}{6} > 3$, a contradiction.

□

Theorem 5. KAL-TOH$_{3,\geq 4}$ *can be answered in linear time.*

Proof. By Lemma 5, the answer to KAL-TOH$_{3,\geq 4}$ is YES if and only if G contains one of the graphs in Figure 2.7 as a subgraph; rephrased using the contrapositive, we see that the answer is NO if and only if the components of G consist only of isolated vertices, single edges, and 2-paths. Thus we can answer this question in $O(n)$ time because computing all components of G and checking each one can be accomplished in $O(n)$ time. □

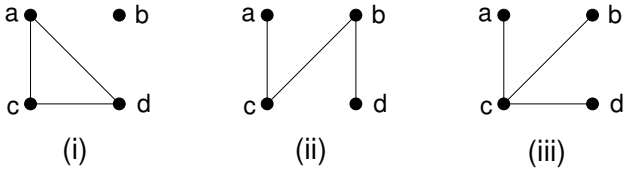

Figure 2.7: The answer to KAL-TOH$_{3,\geq 4}$ is YES if and only if G contains any of these graphs as a subgraph.

Unfortunately, we are unable to use the same reasoning for the problem KAL-TOH$_{2,\geq 4}$. As seen in Figure 2.8, A and B are the only two possible subgraphs of K_4 which contain $6 - 2 = 4$ edges. If a graph G contains either of these as subgraphs, then the answer to KAL-TOH$_{2,\geq 4}$ is YES. However, the claim does not hold in the opposite direction. For a counterexample, consider the dodecahedron having any two of its edges removed; see Figure 2.8. The answer to KAL-TOH$_{2,\geq 4}$ will be YES for this graph, as the dodecahedron (which is triconnected and planar) can be formed by adding the two missing edges, but this graph contains neither A nor B as a subgraph. It is not immediately obvious how many graphs are in the set of "desirable" subgraphs for KAL-TOH$_{2,\geq 4}$ (or even if a finite set of such graphs exists), but we do know that all three graphs seen in Figure 2.8 must be elements of such a set.

However, even if we were able to demonstrate a complete set of "desirable" subgraphs for KAL-TOH$_{2,\geq 4}$, it is still unlikely that this problem could be solved in polynomial time. Checking the existence of any of these subgraphs is equivalent to the SUBGRAPHISOMORPHISM problem, which is NP-complete in general [11]. Hence we conjecture that KAL-TOH$_{2,\geq 4}$ (and similarly KAL-TOH$_{1,\geq 4}$) are NP-hard.

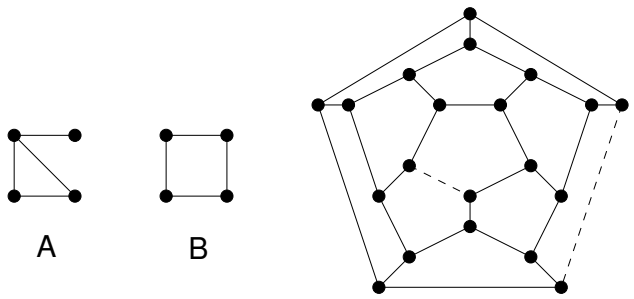

Figure 2.8: The answer to KAL-TOH$_{2,\geq 4}$ is YES if G contains A or B as a subgraph. But the reverse is not true; the dodecahedron having any two of its edges removed serves as a counterexample.

2.2.2 Creating a Prism

Here we show an upper bound on l for the problem KAL-TOH$_{l,=k}$, in which we would like to form a subgraph on exactly $k \geq 4$ vertices after adding at most l edges. This bound is given in Lemma 7 by creating a $\frac{k}{2}$-prism. A *k-prism* is defined as the graph composed of two disjoint cycles $v_1, v_2, \ldots, v_k$ and $w_1, w_2, \ldots, w_k$ where the edge (v_i, w_i) is added for all i.

We first look at the auxiliary result given in Lemma 6.

Lemma 6. *For any $k \geq 4$, there is a graph G on k vertices that is triconnected, planar, 3-regular, and contains a triangle.*

Proof. For any even value of k, consider the $(\frac{k}{2}+1)$-prism H, which is a triconnected, planar, and 3-regular graph on $k+2$ vertices. We label the vertices of H such that the prism is composed of the two cycles $v_1, v_2, \ldots, v_{\frac{k}{2}+1}$ and $w_1, w_2, \ldots, w_{\frac{k}{2}+1}$ where v_i and w_i are adjacent for all i. By contracting edges (v_1, w_1) and (v_2, w_2) in H, we form a graph G on k vertices which is triconnected, planar, 3-regular, and contains a triangle. For any odd value of k, consider the $(\frac{k+1}{2})$-prism H. The argument is similar, except that only a single edge is contracted in H. See Figure 2.9 for an illustration. □

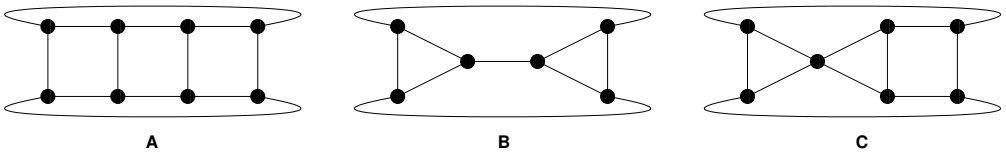

Figure 2.9: Graphs B and C are triconnected, planar, 3-regular, and contain a triangle. B can be created from graph A by contracting two edges, and C can be formed from A by contracting a single edge.

Lemma 7. *The problem* KAL-TOH$_{\frac{3}{2}k-3,=k}$ *(where k is even) [*KAL-TOH$_{\frac{3k+1}{2}-3,=k}$ *(where k is odd)] can be answered in linear time.*

Proof. We first show that KAL-TOH$_{\frac{3}{2}k-3,=k}$ (where k is even) [KAL-TOH$_{\frac{3k+1}{2}-3,=k}$ (where k is odd)] has an answer of YES if and only if G contains at least three edges with at most k distinct endpoints.

Figure 2.10 shows the possible ways in which three edges on at most k vertices (i.e., on at most k distinct endpoints) can be present in G; graphs (i)-(iii) show the possibilities for $k \leq 4$, graphs (i)-(iv) show the possibilities for $k = 5$, and graphs (i)-(v) show the possibilities for $k \geq 6$. Assume that the answer to KAL-TOH$_{\frac{3}{2}k-3,=k}$ is YES for some graph G. Let H be the triconnected planar subgraph created by the addition of edges to G. G must contain at least three edges among the vertices that define H. Since these edges must be part of the graph H on k vertices, then G must initially contain three edges with at most k distinct endpoints.

To show the other direction, assume that G has at least three edges e_1, e_2, and e_3 having at most k distinct endpoints. First consider the case where k is even. If G contains a triangle, then we can add $\frac{3}{2}k - 3$ edges to create the graph described in Lemma 6. Otherwise, G contains one of the configurations (ii)-(v) in Figure 2.10, and we can build a $\frac{k}{2}$-prism (if k is even) or the graph C from Figure 2.9 by adding $\frac{k}{2} - 3$ edges to it.

We have now shown that KAL-TOH$_{\frac{3}{2}k-3,=k}$ (where k is even) [KAL-TOH$_{\frac{3k+1}{2}-3,=k}$ (where k is odd)] has an answer of YES if and only if G contains at least three edges with at most k distinct endpoints. Graphs (i)-(iii) from Figure 2.10 do not appear as subgraphs of G if and only if all components of G are single edges or two-paths. Thus we can answer this question in $O(n)$ time because computing all components of G and checking each one can be accomplished in $O(n)$ time.

□

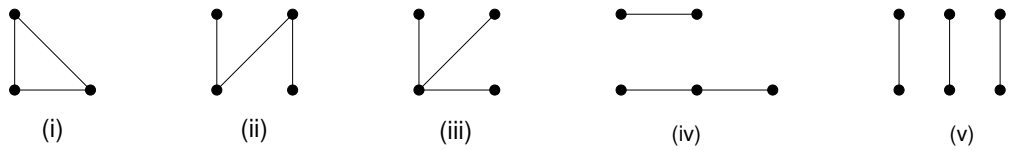

Figure 2.10: Graphs on three edges.

2.2.3 Creating triconnected planar subgraphs of connected graphs

This section shows a second upper bound for l by using a different approach. For a given connected graph G on n vertices, we show that it is always possible to create a triconnected

planar subgraph H of G such that H has exactly $k \leq n$ vertices by adding at most $k+1$ edges.

We begin by finding any spanning tree T of G and then arbitrarily removing exactly $n-k$ leaves. This will leave us with a tree on exactly k vertices such that T is still a subgraph of G. If T contains no vertices of degree two, then we add edges to connect the leaves of T in a cycle to form the graph H. Such a graph is called a *Halin graph*, which is known to be planar and triconnected [37]. An example of a Halin graph is shown in Figure 2.11. By $k \geq 4$, the tree T has at most $k-1$ leaves (which is tight if T is a star), and so it requires the addition of at most $k-1 < k$ edges.

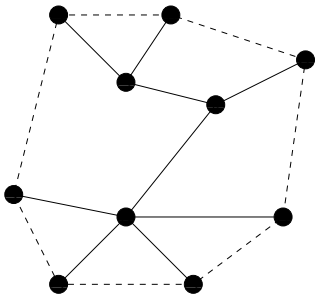

Figure 2.11: A Halin graph.

For the remainder of this section, assume that T has at least one vertex of degree two. Connecting the leaves of T will not be sufficient to form a triconnected graph when T contains a vertex of degree two, so we instead describe an alternate approach. An overview is given below, followed by the details of each step.

1. Select a root node for T
2. Remove an extra degree-two vertex from T (if necessary)
3. Pair non-root degree-two vertices in T such that paired vertices are in different subtrees of the root
4. Create paths between non-root degree-two vertex pairs
5. Connect the remaining leaves
6. Undo the changes done in Step 2 (if any)

Step 1: Select a root for T. If T contains exactly one degree-two vertex, then assign this vertex to be the root (in this case, Steps 2, 3, and 4 may be skipped).

Otherwise, our goal is to select a root vertex r of T that will distribute the degree-two vertices "evenly" among its subtrees. Assign a weight of 1 to each vertex of degree two in T

and assign a weight of 0 to all other vertices. We will denote the total weight of the vertices of T as $w(T)$, which is equal to the number of non-root degree-two vertices in T.

An *α-separator* for T is a vertex v such that each component of $T - v$ has weight at most $\alpha|w(T)|$. Using the separator theorem for trees [4], we know that there exists a $\frac{1}{2}$-separator for T of size one[1]. In other words, there exists a vertex r such that each connected component of $T - r$ contains at most half of the degree-two vertices of T. We select r to be the root of T.

For each non-root vertex v, store a reference to the parent, $parent(v)$. For each vertex of degree at least two, store a list of children, $children(v)$. Each edge in the original spanning tree T is given a direction downward from the root in order to distinguish these edges from those to be added later. For all vertices v, we give the list $children(v)$ an arbitrary order. If Step 2 is not applicable, then we fix these orderings; otherwise some additional changes in the orderings may be made in Step 2.

Step 2: Remove an extra degree-two vertex from T.

This step is only performed if there is an odd number of non-root degree-two vertices in T. Consider the subtrees of the root vertex r $(T_1, T_2, \ldots, T_s)$ which contain at least one degree-two vertex. Among these subtrees, pick the subtree T_i having the smallest index and label one degree-two vertex in T_i as x.

Consider the path found by starting at the root vertex r and repeatedly following the leftmost child of each vertex until a leaf is reached; we call this the **leftmost leaf** of the tree T. For reasons that will be explained in Step 6, we now re-arrange the children at each vertex in T so that x is on the path from r to the leftmost leaf[2].

We now create a dummy leaf d and connect it to x such that d is the rightmost child of x (i.e., the last child in the ordered list $children(x)$). Doing so makes $\deg(x) > 2$ and thus leaves us with an even number of non-root degree-two vertices in T, which is assumed in each of the following steps. Step 6 explains how the vertex d is removed without changing the planarity or the triconnectivity of the graph.

If possible, also re-arrange the order of the subtrees of r (excluding the leftmost subtree) so that the number of vertices in the rightmost subtree of r is greater than one.

Step 3: Pair degree-two vertices in T.

In this step, we create disjoint pairs of non-root degree-two vertices in T according to the "matching" process described below. At the end of Step 3, each non-root degree-two vertex u in T will be paired with exactly one other non-root degree-two vertex v in T. It will become clear why vertices have been paired in this manner once we have reached Step 4.

[1]The definition of a separator found in [4] uses a bound of $\frac{2}{3}$. By inspection, their proof of the separator theorem also shows a weight of at most $\frac{1}{2}|w(T)|$ for each component.

[2]Note that x is not *required* to be within the T_i having the smallest index or be on the path from r to the leftmost leaf. These restrictions are only imposed to ensure that d does not become the leftmost leaf of T.

Consider the subtrees of the root vertex $T_1, T_2, \ldots, T_s$. We will denote the weight of a subtree as $w(T_i)$, which corresponds to the current number of unmatched non-root degree-two vertices in T_i. Note that this weight value may change as non-root degree-two vertices are matched.

In a subtree T_i, we use the term **leftmost degree-two vertex** to denote the first unmatched non-root degree-two vertex found in a depth-first traversal of T_i such that the leftmost unvisited child is selected at each step. The **rightmost degree-two vertex** is defined similarly using a depth-first traversal such that the rightmost unvisited child is selected at each step. A pairing is always made between the rightmost degree-two vertex in a subtree T_i and the leftmost degree-two vertex in a subtree T_j such that $i < j$.

Repeat the following procedure (see Figure 2.12 for an illustration): If all subtrees T_i have a weight of zero, stop. Otherwise, there are still $\alpha \geq 2$ unmatched non-root degree-two vertices remaining among at least two subtrees. Select the smallest index q such that $w(T_1) + w(T_2) + \cdots + w(T_q) > \frac{1}{2}\alpha$. Let $i^* < q$ be the index such that $w(T_{i^*}) > 0$ but $w(T_{i^*+1}), \ldots, w(T_{q-1}) = 0$ (we will soon show that i^* exists). Pair the rightmost vertex of T_{i^*} with the leftmost vertex of T_q, update the weights of T_{i^*} and T_q, and repeat.

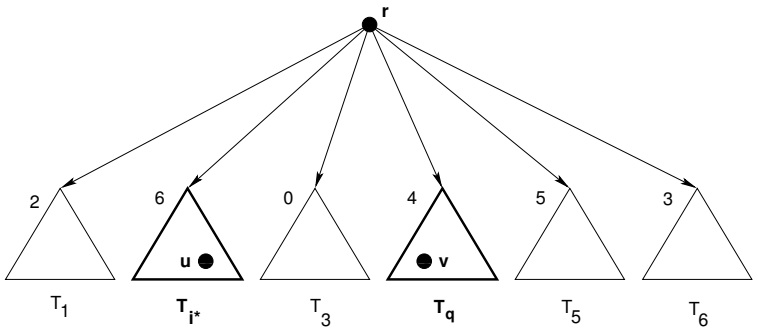

Figure 2.12: (Step 3) The degree-two vertex $u \in T_{i^*}$ is paired with the degree-two vertex v in T_q. The number next to each subtree T_i indicates its weight (i.e., the number of degree-two vertices in T_i).

After the Ith iteration of the above procedure has finished, let $w_I(T_i)$ be the weight of the subtree T_i and let α_I be the sum of the weights of all subtrees T_i, $1 \leq i \leq s$. The proof of Lemma 8 verifies that our procedure matches non-root degree-two vertices in T and terminates only when $\alpha = 0$.

Lemma 8. *$w_I(T_i) \leq \frac{\alpha_I}{2}$ for all iterations I of the above procedure.*

Proof. We show this by induction on I, the iteration number of the procedure. Consider the base case of $I = 0$ (i.e., before the first iteration of the procedure executes). We have that $w_0(T_i) \leq \frac{\alpha_0}{2}$ by our choice of the root node r for T.

For the inductive hypothesis, assume that $w_I(T_i) \leq \frac{\alpha_I}{2}$ holds for all $1 \leq i \leq s$ for iterations $1, 2, \ldots, I$. We now show that $w_{I+1}(T_i) \leq \frac{\alpha_{I+1}}{2}$ for all $1 \leq i \leq s$. Note that $\frac{\alpha_{I+1}}{2} = \frac{\alpha_I}{2} - 1$

because exactly two non-root degree-two vertices are matched at each iteration. In the cases below, the index values i refer to the subtrees after the $(I+1)$st iteration.

Case 1 ($i^* < i < q$). For all index values of i in this range, $w_{I+1}(T_i) = 0$ by our definition of i^* and q. Since $\alpha_{I+1} \geq 0$, we have that $w_{I+1}(T_i) \leq \frac{\alpha_{I+1}}{2}$.

Case 2 ($i = i^*$ or $i = q$). A non-root degree-two vertex in T_{i^*} is matched with a non-root degree-two vertex in T_q at each iteration, thus we have $w_{I+1}(T_i) < w_I(T_i)$ when $i = i^*$ or $i = q$. By the inductive hypothesis, $w_I(T_i) \leq \frac{\alpha_I}{2}$ and thus $w_{I+1}(T_i) \leq \frac{\alpha_I}{2} - 1 = \frac{\alpha_{I+1}}{2}$.

Case 3 ($i < i^*$). We have $w_I(T_i) < \frac{\alpha_I}{2}$ since $w(T_{i^*}) \leq \frac{\alpha_I}{2}$ and $w(T_{i^*}) > 0$. This gives $w_{I+1}(T_i) \leq w_I(T_i) \leq \frac{\alpha_I}{2} - 1 = \frac{\alpha_{I+1}}{2}$.

Case 4 ($i > q$). By our definition of q, $\sum_{j=1}^{q} w_I(T_j) > \frac{\alpha_I}{2}$, and so we have $w_I(T_i) \leq \alpha_I - \sum_{j=1}^{q} w_I(T_j) < \frac{\alpha_I}{2}$. This gives $w_{I+1}(T_i) \leq w_I(T_i) \leq \frac{\alpha_I}{2} - 1 = \frac{\alpha_{I+1}}{2}$. □

Note that $w(T_q) > 0$ at each iteration, otherwise a smaller index would have been selected for q. Also, $w(T_q) \leq \frac{\alpha}{2}$, which implies that T_q cannot have been the first subtree with positive weight, so i^* exists. We conclude that the above procedure will always match a pair of non-root degree-two vertices in different subtrees at every step, and therefore no unmatched degree-two vertex will remain when the procedure terminates.

Step 4: Create paths between non-root degree-two vertex pairs.

We now use the pairings found in Step 3 in order to add edges to T.

Let $u \in T_i$ and $v \in T_j$ be two paired vertices such that $i < j$. Consider the unique path P from u to the root of T using only the directed edges in the original spanning tree. Let S_u be the set of all leaves that are descendants of any vertex in P (excluding the root) *and* that appear to the right of the path P in T_i. S_v is defined similarly, but with the leaves appearing to the left of the path instead. Note that we only select leaves for S_u and S_v which have *not* already been connected by an added edge (i.e., the vertices which still have degree 1).

Add undirected edges between the leaves in S_u and S_v while maintaining their left-to-right ordering to form an undirected path between u and v; see Figure 2.13. Note that the edges added in this step are undirected to distinguish them from the directed edges, which existed in the original spanning tree T.

The pairing procedure from Step 3 guarantees that the edge additions in Step 4 will maintain the planarity of T. This is because, for any pairing between the rightmost degree-two vertex $u \in T_i$ and the leftmost degree-two vertex $v \in T_j$, $i < j$, we have the property that $w'(T_l) = 0$ for all $i < l < j$ (i.e. T_l would not be involved in any further pairings). The only vertices possibly removed from the outer-face are vertices that are between u and v in the left-first search. These vertices have degree at least three because they either had degree at least three in the original spanning tree or they were leaves that have already been connected with two added edges. Hence the addition of edges between a non-root degree-two vertex in T_i and a non-root degree-two vertex in T_j can always be made such that no crossings would be forced.

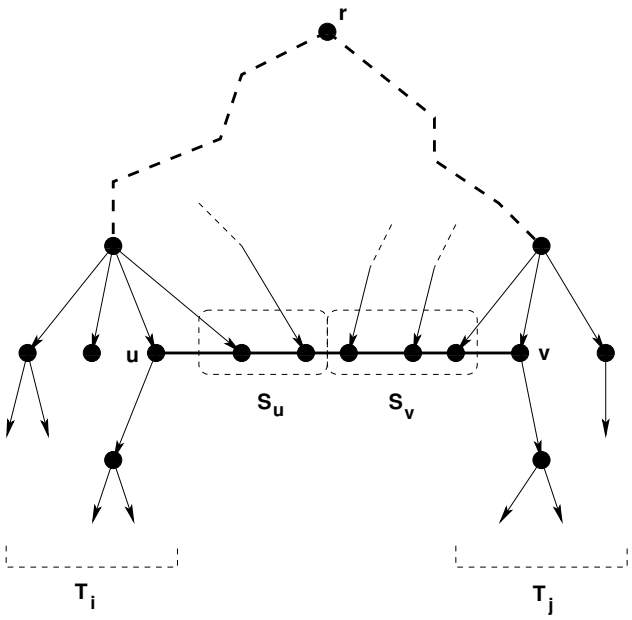

Figure 2.13: (Step 4) An undirected path is formed between $u \in T_i$, the leaves in S_u, the leaves in S_v, and $v \in T_j$. The non-root degree-two vertices u and v were paired in Step 2.

Step 5: Connect the remaining leaves.

We argue that there must be at least two leaves remaining in T at the end of Step 4. If we did no pairing, then T must have at least two leaves because $deg(r) \geq 2$. Otherwise, consider the final pairing of non-root degree-two vertices u and v from Step 4. There must be a non-empty child of each of these two vertices, which is either a leaf or a vertex of degree at least three, which must have a leaf as a descendant, so T has at least two leaves.

Connect the remaining leaves in left-to-right order using undirected edges; if there are more than two leaves remaining, we add another edge between the leftmost leaf x_L and rightmost leaf x_R. We also add undirected edges (r, x_L) and (r, x_R), which guarantees that $\deg(r) > 2$. See Figure 2.14 for an illustration.

At the end of Step 5, we have a graph H on k vertices (or $k + 1$ vertices if Step 2 was followed) that was created by adding edges to the tree T.

Step 6: Undo the changes done in Step 2 (if any)

If we added a dummy vertex d in Step 2, we must remove this vertex to preserve the original number of vertices in H. The re-arrangements performed in Step 2 before adding the dummy leaf d guarantee that $d \neq x_L$ and $d \neq x_R$ once the remaining leaves are connected at the end of Step 5, thus $\deg(d) = 3$. Let x, a, and b be the neighbours of d. Using a process known as a $Y\Delta$-transformation (illustrated in Figure 2.15), we replace edges (a, d), (b, d), and (x, d) with the edges (a, b), (a, x), and (b, x). A $Y\Delta$-transformation does not increase

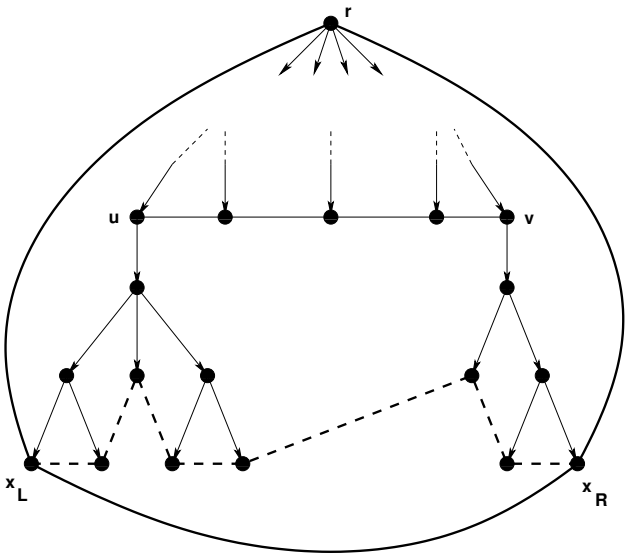

Figure 2.14: (Step 5) The remaining leaves are connected in an undirected path (dashed). The three edges (x_L, x_R), (r, x_L), and (r, x_R) are also added (bolded).

the number of edges in H and also preserves the planarity and triconnectivity of H [42]. The isolated vertex d is then discarded after this transformation.

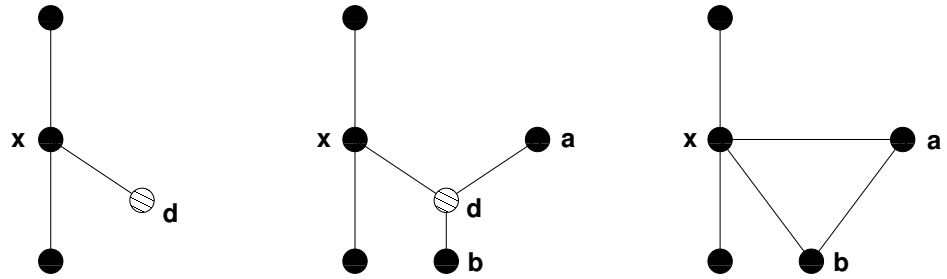

Figure 2.15: The dummy leaf d is connected to the non-root degree-two vertex x in Step 2 so that there are an even number of non-root degree-two vertices in the graph (left). At the end of Step 5, d is connected to two other vertices a and b (centre). A $Y\Delta$-transformation removes the dummy vertex d from H while preserving planarity and triconnectivity (right).

At the end of this section we will justify that the graph H is triconnected and planar by the end of Step 5; we call this graph H_5. Hence after Step 6 (i.e., the $Y\Delta$-transformation) the resulting graph is triconnected and planar.

This ends the description of our six steps involved in our procedure to triconnect T while maintaining planarity. We now make the following definitions:

Definition 6 (leftmost path, rightmost path)**.** *Consider the graph H_5 formed by adding undirected edges to T up to the end of Step 5 in our procedure. Let v be any non-root vertex with incoming directed edge e_0 such that the counterclockwise ordering of edges incident to v, beginning with edge e_0, is $C = (e_0, e_1, e_2, \ldots, e_0)$. The* ***leftmost path*** *from v to x_L in H_5 is found by repeatedly following edge e_1 in this ordering. The* ***rightmost path*** *from v to x_R in H_5 is defined similarly, but instead using "clockwise order."*

Let e be any edge in H_5. We assign a label of 1 to e if there exists a vertex v such that e is on the unique directed path from v to r, the root of the original spanning tree T (as it appears at the end of Step 2). In other words, e is assigned a label of 1 if it was an edge in the original spanning tree. The undirected edges of H_5 do not receive a label of 1. A label of 2 is assigned to an edge e if there exists a vertex v such that e is on the rightmost path from v to x_R, and a label of 3 is assigned to an edge e if there exists a vertex v such that e is on the leftmost path from v to x_L. Note that some edges receive multiple labels.

Some exceptions to this general rule are that the added edges (r, x_L), (r, x_R), and (x_L, x_R) have labels $(1, 3)$, $(1, 2)$, and $(2, 3)$ respectively. Furthermore, any directed edges incident to x_L or x_R in the original spanning tree do not receive a label of 1. Figure 2.16 illustrates the labeling of edges incident to each type of vertex in H_5. As argued in Lemmas 9 and 10, every edge label contains either one or two values.

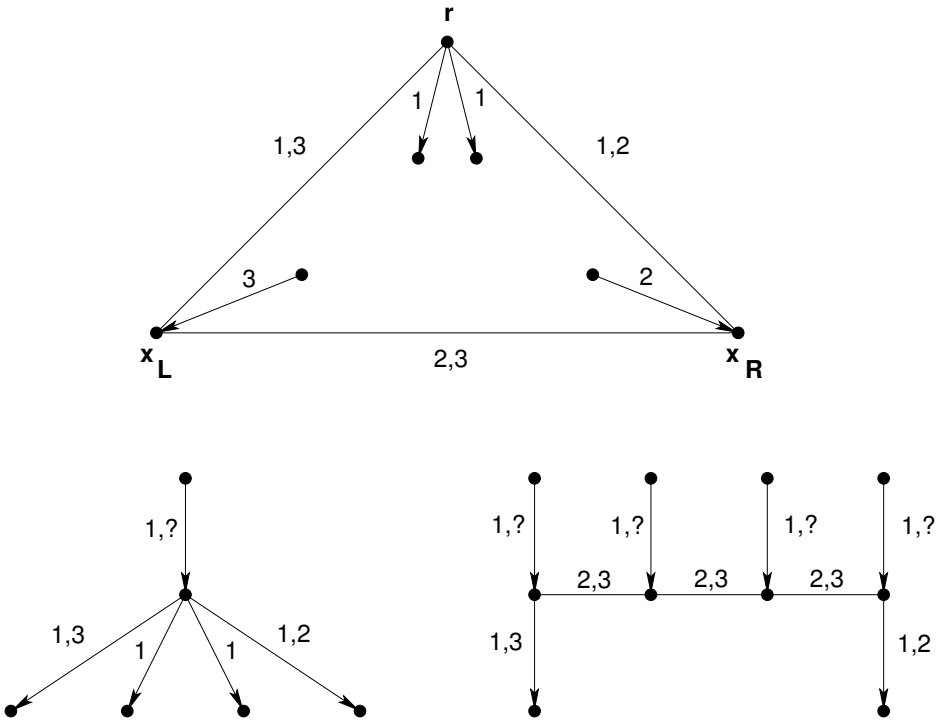

Figure 2.16: Adding labels to the edges of H_5.

Lemma 9. *Each edge in H_5 contains at least one of the labels 1, 2, or 3.*

Proof. By our description of the labeling procedure, we know that all directed edges contain the label of 1 because they were part of the original spanning tree. We also know that any edge incident with r, x_R, and x_L contains a label of 1, 2, and 3, respectively. This follows from our description of the labeling procedure.

The remaining edges (v, w) to consider are undirected and were added during Steps 4 and 5 of our procedure. Each of v and w was either a leaf (not equal to x_L or x_R) or a non-root degree-two vertex in the original spanning tree. By our definitions of leftmost and rightmost paths in H_5, we see that each edge (v, w) contains labels 2 and 3. Therefore each edge in H_5 contains at least one label. □

Lemma 10. *No edge in H_5 may contain all of the labels 1, 2, and 3.*

Proof. By our description of the labeling procedure, all edges incident with any of r, x_L, or x_R will not have all three labels. Furthermore, any undirected edge in H_5 cannot contain a label of 1 because it was not part of the original spanning tree. So we only need to consider directed edges (v, w) that were part of the original spanning tree such that neither v nor w is r, x_L, or x_R.

Assume that there exists such an edge having all three labels 1, 2, and 3. Since v has an outgoing directed edge, we must have that either $\deg(v) = 2$ or $\deg(v) > 2$ in the original spanning tree. If $\deg(v) = 2$ in the original spanning tree, then v must be incident to exactly one undirected edge e. Thus (v, w) cannot contain both labels 2 and 3 because either the rightmost path or the leftmost path originating from v must use edge e instead of (v, w). If $\deg(v) > 2$ in the original spanning tree, then it is impossible for both the leftmost path and the rightmost path originating from v to use the edge (v, w), thus (v, w) cannot contain both labels 2 and 3. □

We use the notation $(1, ?)$ to indicate that the labeling of an edge is one of (1), $(1, 2)$, or $(1, 3)$. Note that a downward edge only has label (1) if it is incident to a vertex v in the original spanning tree having at least three children and if the edge is incident to neither the leftmost nor rightmost child of v. These edges cannot be a part of any leftmost or rightmost paths and therefore cannot have a label of 2 or 3. The notation $(2, ?)$ is used to indicate that the labeling of an edge is one of (2) or $(1, 2)$, and the notation $(3, ?)$ is used to indicate that the labeling of an edge is one of (3) or $(1, 3)$.

A label on a directed edge e incident to vertex v is called an **incoming label** to vertex v if it corresponds to the incoming direction to v on e. An **outgoing label** to vertex v on a directed edge e corresponds to the outgoing direction from v on e. The labels of 2 and 3 on an undirected edge e are labeled as incoming or outgoing relative to an incident vertex v according to the order in which v and e appear on a leftmost or rightmost path.

We will now show that this labeling is a *triorientation* of the edges of H_5, i.e. a *Schnyder wood*. For the remainder of this section, the indices i, $i-1$, and $i+1$ are used $\pmod 3 + 1$.

Definition 7 (Triorientation)**.** *[13] A* ***triorientation*** *of a set of edges replaces each edge with a single directed edge or with two opposite directed edges. Each direction is given one of the labels 1, 2, or 3 and two opposite directions must have distinct labels.*

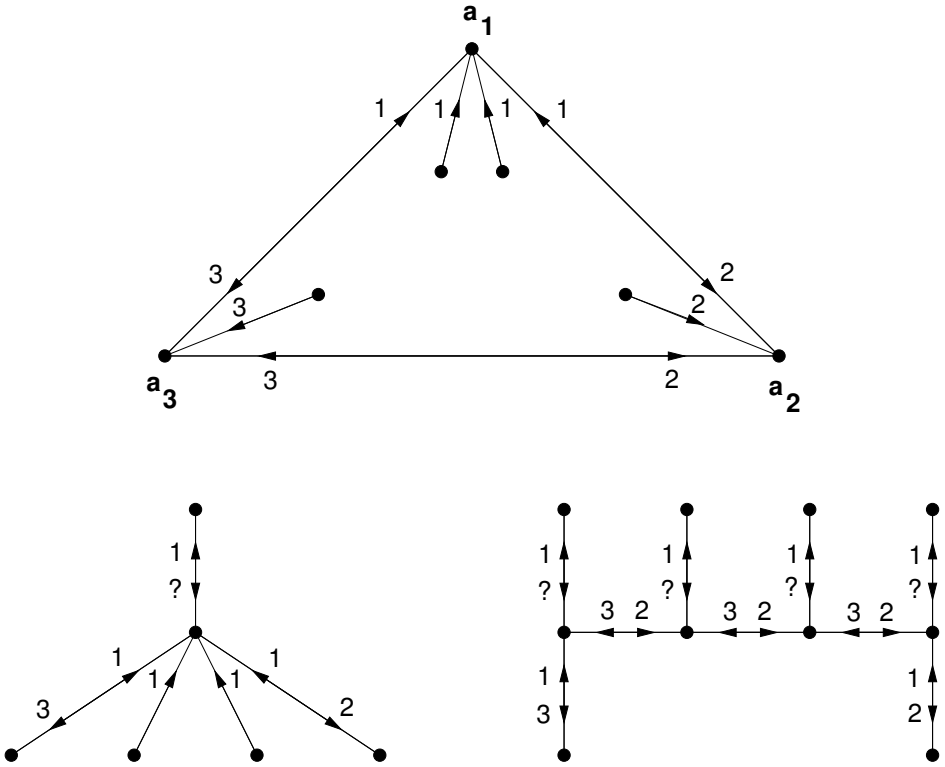

Figure 2.17: A triorientation of the edges of H_5.

Definition 8 (Schnyder wood)**.** *[3] Let G be a planar graph with outer triangle $\{a_1, a_2, a_3\}$. A triorientation of the interior edges of G is a* ***Schnyder wood*** *iff the following conditions are satisfied:*

1. *For $i = 1, 2, 3$, all incoming labels to a_i are i, and a_i has no outgoing edges with label i.*

2. *Every internal vertex v has exactly three outgoing labels 1, 2, and 3 appearing in clockwise order around v.*

3. *If i is an incoming label to an internal vertex v, then it appears between the two outgoing labels $i-1$ and $i+1$ of v.*

The necessary conditions for the edges around an internal vertex in the definition of a Schnyder wood are illustrated in Figure 2.18. By examining all possible types of internal

vertices within the graph as shown in Figure 2.17, we see that these conditions have been satisfied. All incoming labels to a_i are i, so we conclude that our orientation and labeling of the graph is a Schnyder wood.

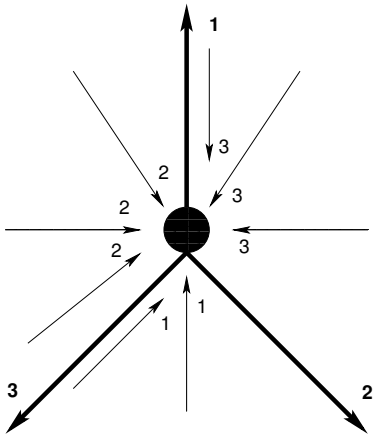

Figure 2.18: The ordering of labels around an internal vertex v of a Schnyder wood. v has exactly three outgoing labels 1, 2, and 3 in clockwise order. All incoming labels i must appear between the outgoing labels $i-1$ and $i+1$.

As we will see (we defer the proof to Section 2.2.4), every graph with a Schnyder wood and with edges (a_1, a_2), (a_2, a_3), and (a_1, a_3) is triconnected. Since H_5 has a Schnyder wood and the edges (x_L, x_R), $(r, x_L,)$, and (r, x_R) exist, we see that H_5 is triconnected. The $Y\Delta$-transformation performed in Step 6 perserves triconnectivity [42], hence the graph that we create by adding edges to T is planar and triconnected. All that remains to do is analyze the number of edges.

Theorem 6. *Given a tree T on $k \geq 4$ vertices, it is possible to create a triconnected planar graph H such that T is a spanning tree of H by adding at most $k+1$ edges to T.*

Proof. If T has no vertices of degree two, then we join its leaves in a cycle to form a Halin graph. T will have a maximum of $k-1$ leaves (if it is a star), so we have added at most $k-1$ edges. So assume that T has vertices of degree two and select a root r as described in Step 1.

For $i = 1, 2$, let k_i be the number of non-root vertices of degree i in T, excluding the root node r. We apply Step 2 only if k_2 is odd, and in this case $k_2' = k_2 - 1$ and $k_1' = k_1 + 1$ are the new bounds on the non-root vertices of degree 2 and 1, respectively, in the resulting tree.

Let $2t$ and l be the number of non-root degree-2 and degree-1 vertices, respectively, after Step 2 has been (perhaps) applied. Adding a path P having l_P leaves in Step 4 will reduce l by l_P and will reduce t by 1. In total, Step 4 uses $l_4 + t$ edges, where l_4 is the number of

leaves processed in Step 4. Thus in Step 5 we have $l - l_4$ leaves remaining and use at most $l - l_4$ edges for the cycle among them.

Note that the two edges (r, x_L) and (r, x_R) are also added to T. So the number of added edges is $l + t + 2$ in Steps 4-5, plus perhaps one more from Step 2, which gives a bound of $k_1 + \frac{k_2}{2} + 2$ if k_2 is even and a bound of $1 + k_1' + \frac{k_2'}{2} + 2 = k_1 + \frac{k_2}{2} + \frac{7}{2}$ if k_2 is odd.

Case 1 ($k_2 = 0$). Here, the root vertex has degree two, otherwise we would have connected the leaves of T to form a Halin graph. We must also have that $k_1 > 2$ because the total number of vertices in T must be at least four. So without loss of generality, assume that the first vertex in $children(r)$ has degree at least three.

If the second vertex in $children(r)$ is a leaf, then k_1 edges will be added to connect the leaves of T in a cycle, (r, x_L) will be added, but the edge (r, x_R) will *not* be added because these two vertices are already adjacent. In this case, $k_1 \leq k - 2$ and so we have added $k_1 + 1 \leq k - 1$ edges to T.

If the second vertex in $children(r)$ is not a leaf, then both (r, x_L) and (r, x_R) will be added to T. In this case, $k_1 \leq k - 3$ and so we have added $k_1 + 2 \leq k - 1$ edges to T.

Case 2 ($k_2 = 1$). In the special case where $\deg(r) = 2$ and $k_1 = 2$, then T is a path on four vertices. The addition of exactly three edges is necessary and sufficient to triconnect T (which will form K_4), and because our procedure does not add multiple edges between any two vertices, we have added a total of $3 = k - 1$ edges. So for the remainder of this case we will assume that $k_1 > 2$ and $\deg(r) > 2$.

When $k_2 = 1$, no edges are added in Step 4 and k_1' edges are added in Step 5 to connect all leaves in a cycle, and $k_1' = k_1 + 1$. Another edge is also added in Step 2 when connecting d to the single non-root degree-two vertex x.

Step 2 re-arranged the subtrees of the root vertex so that x is in the leftmost subtree and the dummy vertex d does not become either x_L or x_R. This also implies that x is not the rightmost child of r. Furthermore, the subtrees of the root vertex were re-arranged in Step 2 so that, if possible, the rightmost subtree has more than one vertex. If so, r would not be adjacent to the leaf x_R at the end of Step 5.

If the rightmost subtree of r is the leaf x_R, then the undirected edge (r, x_R) is not added, but the edge (r, x_L) is still added. In this case, $k_1 \leq k - 2$ and so we have added $k_1' + 2 = k_1 + 3 \leq k + 1$ edges to T.

If the rightmost subtree of r is not the leaf x_R, then both undirected edges (r, x_R) and (r, x_L) are added. In this case, $k_1 \leq k - 3$ and so we have added $k_1' + 3 = k_1 + 4 \leq k + 1$ edges to T.

Case 3 ($k_2 = 2$). We will have added at most $k_1 + 3$ edges to T. Since $k_1 \geq 2$ and the root node has not been counted, then $k_1 + 3 \leq (k - 3) + 3 = k$. So we have added at most k edges to T in this case.

Case 4 ($k_2 = 3$). $l_4 + 1$ edges are added to T in Step 4, $k_1' - l_4$ edges are added to Step 5 to connect the remaining leaves in a cycle, one edge is added from Step 2, and two edges are

added to connect r with x_L and x_R. This gives a total of $(l_4+1)+(k'_1-l_4)+3 = k'_1+4 = k_1+5$ edges added to T. Here we see that $k_1 \leq k-4$, which means that we have added at most $k+1$ edges to T.

Case 5 ($k_2 = 2t \geq 4$). If the number of non-root vertices of degree two is even, then we will have added a total of at most $k_1 + \frac{k_2}{2} + 2$ edges to T. First observe that $\frac{k_2}{2} + 2 \leq k_2$ and, because the root node is not counted as part of k_1 or k_2, we have that $k_1 + k_2 \leq k-1$. Combining these inequalities gives $k_1 + \frac{k_2}{2} + 2 \leq k-1$, so we have added at most $k-1$ edges to T.

Case 6 ($k_2 = 2t+1 \geq 5$). If the number of non-root degree-two vertices is odd, then Step 2 adds one degree-one vertex and adds one to the degree of a single degree-two vertex, so $k'_2 = k_2 - 1$, $k'_1 = k_1 + 1$, and $k' = k+1$. The $Y\Delta$-transformation in Step 6 neither increases nor decreases the number of edges in the graph, but we must still count the addition of the single edge made in Step 2.

Steps 2, 4, and 5 add $1 + k'_1 + \frac{k'_2}{2} + 2 = \left(k_1 + \frac{k_2}{2} + 2\right) + \frac{1}{2} \leq (k-1) + \frac{1}{2} = k - \frac{1}{2}$ edges (the inequality holds by the same argument presented where $k_2 \geq 4$ and was even). But since this value must be an integer, we have that the number of added edges is at most $k-1$.

□

Theorem 6 implies the two Corollaries listed below.

Corollary 2. *Any connected graph G on at least k vertices contains a triconnected subgraph H on exactly k vertices after the addition of at most $k+1$ edges.*

Corollary 3. *The answer to* Kal-Toh$_{k+1,=k}(G)$ *is* Yes *for all $k \geq 4$ if G is connected.*

From our discussion of Halin graphs, we see that $k-1$ is a lower bound for the minimum number of edges needed to triconnect a tree on k vertices. Notice that in most cases of the proof of Theorem 6, the bound achieved is actually $k-1$; the bound of k or $k+1$ occurs only if $k_2 \leq 3$. It seems quite likely that these special situations could be treated a bit differently to match the lower bound of $k-1$ in all cases. We leave this for future study.

2.2.4 Schnyder woods and triconnectivity

It is known that if G is a triconnected graph, then G has a triorientation which corresponds to a Schnyder wood [13], [3]. Here we give a proof of the reverse, which is a vital ingredient for Theorem 6. According to S. Felsner (private communication) this is "quite obvious," but we have not been able to find a formal proof and hence provide one here.

Theorem 7. *If a graph G has a Schnyder wood and contains the edges (a_1, a_2), (a_2, a_3), and (a_1, a_3), then it is triconnected.*

Before giving the proof, we need some notation and a few observations.

A triorientation of the edges of G defines the three directed trees D_1, D_2, and D_3; the tree D_i contains exactly the directed edges from G having label i. Each tree D_i is rooted at a_i and includes all vertices of G. It follows from the definition of a Schnyder wood that a_i has outdegree 0 and all other vertices have outdegree 1 in D_i.

Let $P_i(v)$ denote the unique directed path in D_i from a vertex v to a_i.[3] Clearly each D_i is acyclic. As argued in [13], $D_i \cup D_{i-1}^{-1} \cup D_{i+1}^{-1}$ is also acyclic, where D_j^{-1} denotes the tree D_j having the orientation of its edges reversed.

The following three Lemmas (11, 12, and 13) about properties of Schnyder woods were given in [13]. The proofs of Lemmas 11 and 13 were left to the reader in [13]; we show them here to demonstrate that they do not rely on triconnectivity.

Lemma 11. *For all vertices v in H, the paths $P_i(v)$ and $P_j(v)$, $i \neq j$, have only the vertex v in common.*

Proof. Assume the opposite. Let $w \neq v$ be a vertex on both $P_i(v)$ and $P_j(v)$. Then there exists a directed cycle containing both v and w in the graph $D_i \cup D_j^{-1}$, which is a contradiction because each D_i is acyclic. □

For any vertex v, Lemma 11 implies that the three paths $P_1(v)$, $P_2(v)$, and $P_3(v)$ have only the vertex v in common. These paths therefore divide G into three regions $R_1(v)$, $R_2(v)$, and $R_3(v)$ where $R_i(v)$ is the region bounded by (and including) the path $P_{i-1}(v)$, the path $P_{i+1}(v)$, and the edge (a_{i-1}, a_{i+1}) (see Figure 2.19). Denote the *open region* $R_i^o(v)$ to be $R_i(v) - (P_{i-1}(v) \cup P_{i+1}(v))$, i.e. $R_i^o(v)$ does not include its bounding paths.

Lemma 12. *If v and w are two vertices of a Schnyder wood with $w \in R_i(v)$, then $R_i(w) \subseteq R_i(v)$. If $w \in R_i^o(v)$, then the inclusion is proper: $R_i(w) \subset R_i(v)$.*

Proof. See [13]. By inspection of the proof given there, one sees that it does not rely on triconnectivity. □

Lemma 13. *Let G be a Schnyder wood, v a vertex in G, and w a vertex in $R_i^o(v)$. If x is the first vertex on $P_{i+1}(w)$ incident with $P_{i-1}(v) \cup P_{i+1}(v)$, then x is on $P_{i+1}(v)$ and $x \neq v$. Similarly, if x is the first vertex on $P_{i-1}(w)$ incident with $P_{i-1}(v) \cup P_{i+1}(v)$, then x is on $P_{i-1}(v)$ and $x \neq v$.*

Proof. Consider the first statement of the lemma. Without loss of generality, assume $i = 1$, so w is a vertex in $R_1^o(v)$, and x is the first vertex on $P_2(w)$ incident with $P_2(v) \cup P_3(v)$. For any vertex on the path $P_3(v)$, the incoming labels of 2 are from outside region $R_1(v)$ or on the

[3]Note that in our construction for Theorem 6, $P_1(v)$ corresponds to the path from v to the root r in the original spanning tree T. Furthermore, $P_2(v)$ corresponds to the rightmost path from v in H and $P_3(v)$ corresponds to the leftmost path from v in H.

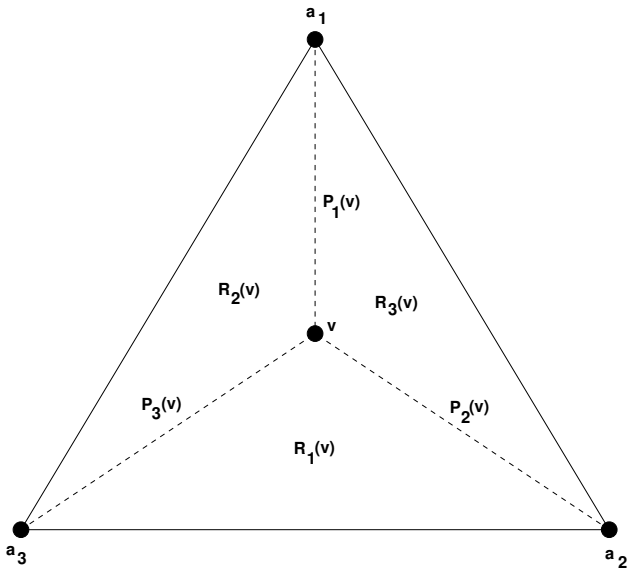

Figure 2.19: The paths from v to a_1, a_2, and a_3 divide H into three regions.

path $P_1(v)$ by the order of edges; see also Figure 2.20. So if x were on $P_3(v)$, its predecessor on path $P_2(w)$ would have to be outside $R_1(v)$, contradicting planarity. Therefore x must be on $P_2(v) - \{v\}$. The second statement of the lemma is shown in a similar manner.

□

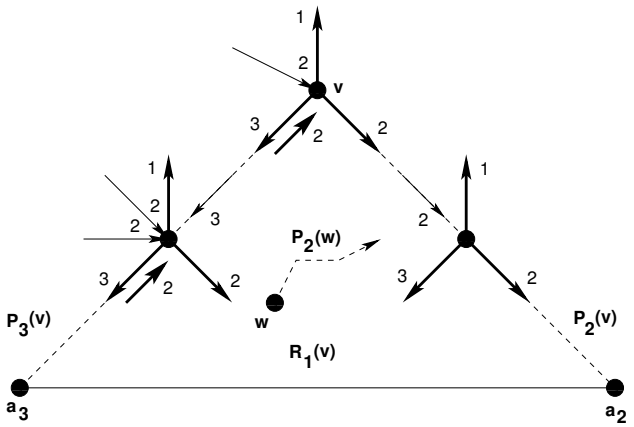

Figure 2.20: The possible edge orientations for vertices on $P_2(v) \cup P_3(v)$.

Lemmas 11, 12, and 13 are now used in order to show Theorem 7, i.e. the graph G is triconnected. To do so, let v and w be two vertices of G. We will show that there exist three vertex-disjoint paths between v and w. We consider three cases, which are illustrated in Figure 2.21. By symmetry, we may assume that $w \in R_1(v)$.

Case 1: $w \in R_1^o(v)$.

Let x be the first vertex on $P_2(w)$ that is incident with the path $P_2(v) \cup P_3(v)$. By Lemma 13, x must be on the path $P_2(v)$ and $x \neq v$. Define the vertex y to be the first vertex on the path $P_3(w)$ that is incident with the path $P_2(v) \cup P_3(v)$; we have that y is on $P_3(v)$. Now consider the path $P_1(w)$. Since $P_1(w) \subsetneq P_1(v)$ by Lemma 12, and since $P_1(w)$, $P_2(w)$, and $P_3(w)$ have only the vertex w in common, $P_1(w)$ intersects $P_2(v) \cup P_3(v)$ at a vertex z in the interior of the path $y \overset{P_3^{-1}(v)}{\rightsquigarrow} v \overset{P_2(v)}{\rightsquigarrow} x$.

If $z \in P_3(v)$, we have the paths $v \overset{P_1(v)}{\rightsquigarrow} a_1 \to a_3 \overset{P_3^{-1}(w)}{\rightsquigarrow} y \overset{P_3^{-1}(w)}{\rightsquigarrow} w$, $v \overset{P_3(v)}{\rightsquigarrow} z \overset{P_1^{-1}(w)}{\rightsquigarrow} w$, and $v \overset{P_2(v)}{\rightsquigarrow} x \overset{P_2^{-1}(w)}{\rightsquigarrow} w$. The case where $z \in P_2(v)$ is symmetric.

Case 2: $w \in P_3(v)$.

Let u be the vertex adjacent to w on the path from v to w along $P_3(v)$. We now consider two subcases based on the labels of the edge (u, w).

Case 2(a): The edge (u, w) does not contain a label of 1.

Because the edge (u, w) does not contain a label of 1, the successor q of w on path $P_1(w)$ is in $R_2^o(v)$. Let z be the first vertex incident with $P_1(v)$ when following the path $P_1(w)$ from q. Note that $z \neq v$ because otherwise $v \in P_2(w) \cap P_1(w)$, contradicting Lemma 11.

We have the three disjoint vw-paths $v \overset{P_3(w)}{\rightsquigarrow} u \to w$, $v \overset{P_1(v)}{\rightsquigarrow} z \overset{P_1^{-1}(w)}{\rightsquigarrow} q \to w$, and $v \overset{P_2(w)}{\rightsquigarrow} a_2 \to a_3 \overset{P_3^{-1}(w)}{\rightsquigarrow} w$.

Case 2(b): The edge (u, w) contains a label of 1.

Because (u, w) lies on the path $P_3(v)$, then it also contains a label of 3 and thus cannot contain a label of 2 by Lemma 10. Then the successor q of w on path $P_2(w)$ is in $R_1^o(v)$. Let z be the first vertex incident with $P_2(v)$ when following the path $P_2(w)$ from q. Note that $z \neq v$ because otherwise $v \in P_2(w) \cap P_3(w)$, contradicting Lemma 11.

We have the three disjoint vw-paths $v \overset{P_3(w)}{\rightsquigarrow} u \to w$, $v \overset{P_1(v)}{\rightsquigarrow} a_1 \to a_3 \overset{P_3^{-1}(w)}{\rightsquigarrow} w$, and $v \overset{P_2(v)}{\rightsquigarrow} z \overset{P_2^{-1}(w)}{\rightsquigarrow} q \to w$.

Note that $w \in P_2(v)$ is symmetric to Case 2. This ends the proof that G is triconnected, and hence finishes the proof of Theorem 6 as well.

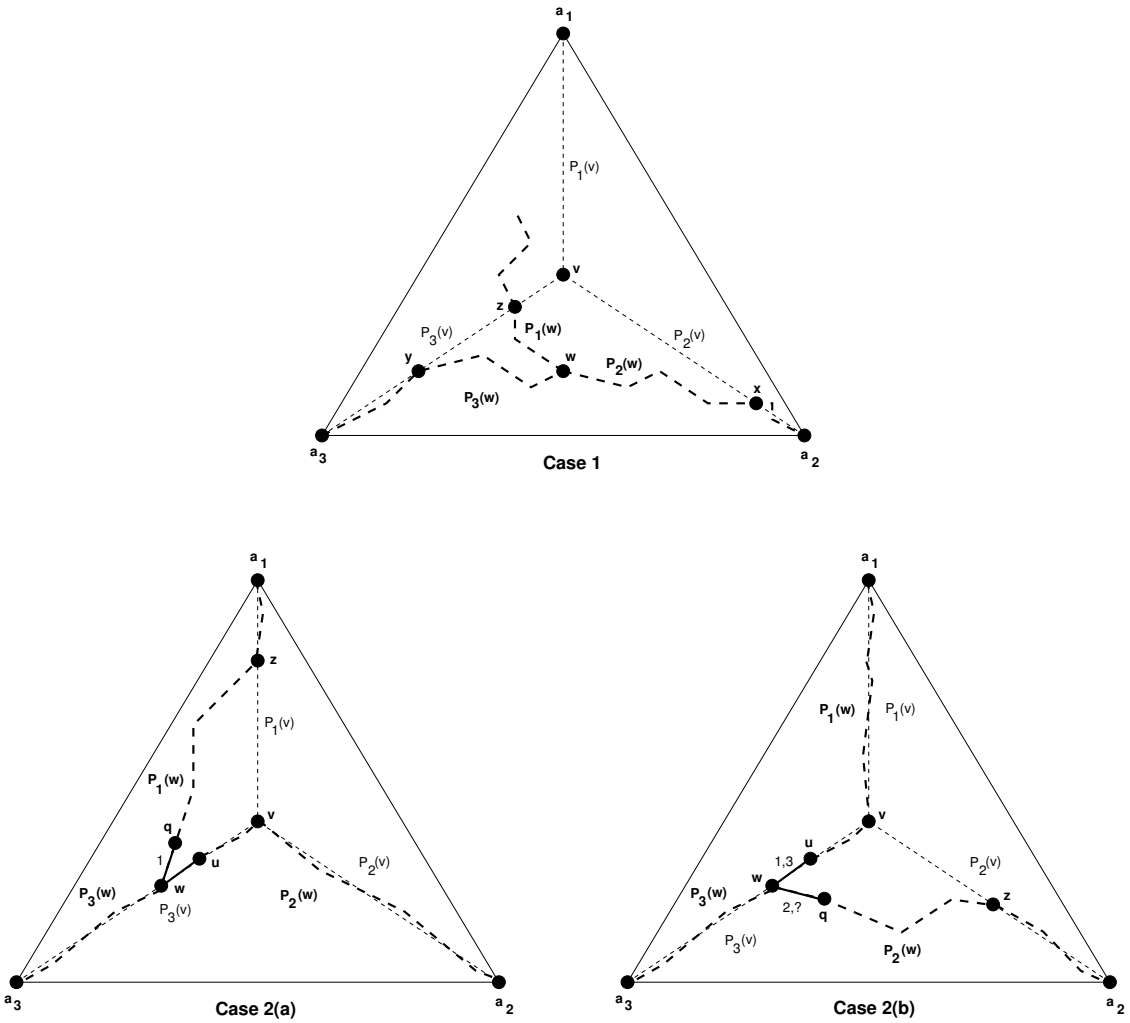

Figure 2.21: Illustrations for the cases in the proof that G is triconnected.

Chapter 3

Kal-toh for Two Players

In this chapter, we give a formulation of the two-player version of Kal-toh. We had hoped that we could find a simple, Nim-like condition to determine in $O(1)$ time which player will win the game. Unfortunately, for all but the most trivial target graphs H this remains elusive. This chapter details some of the approaches that we have found, and should be viewed as the beginning of an exploration into solving Kal-toh for two players, as many questions remain open.

3.1 Formulating the Game

Recall our interpretation of the one-player version of Kal-toh as a graph problem, studied in Chapter 2:

KAL-TOH$_{l,\geq k}$ Let G be a graph on a set of $n \geq 4$ vertices. Does there exist a triconnected planar subgraph H of G on at least $4 \leq k \leq n$ vertices after the addition of at most l edges?

We will give a similar interpretation for the two-player version of Kal-toh. The game begins with a graph G on a set of $n \geq 4$ vertices. Unless stated otherwise, it is assumed that G initially has no edges. Two players take turns by alternately adding an edge between any two vertices of G; multiple edges are not allowed. Note that this ignores any restrictions on edge length or distance between vertices that may arise when the game is played in three dimensions with metal rods.

The analysis of the game becomes more complicated with two players. If the winning subgraph H can be *any* triconnected planar graph, then a player would need to consider the answer to all of the following questions on each turn:

- Can an edge e be added such that $G + e$ contains a triconnected planar subgraph H?

- If not, which edges e may be added such that $(G+e)+e'$ does not contain a triconnected planar subgraph H for any edge e' not in $G+e$?

Answering these questions is likely NP-hard based on our conjectures in Chapter 2, so we make a change to the game model in hopes of finding a polynomial-time solution for a game instance. Unlike the one-player variant, we specify a target subgraph H in the two-player version of Kal-toh such that the first player to create H as a subgraph of G is declared the winner. In other words, the player who adds edge e such that $G+e$ contains H as a subgraph, but G does *not* contain H as a subgraph, will win the game.

We therefore phrase the two-player version of Kal-toh in the following way:

KAL-TOH-2P$_n(H)$ Starting with the empty graph on $n \geq 4$ vertices and target subgraph H, which player has a winning strategy for the two-player game of Kal-toh?

3.1.1 Kal-toh as an impartial game

KAL-TOH-2P$_n(H)$ is an *impartial* game, which means that either player would be able to make exactly the same possible moves on any game configuration regardless of whose turn it is to act. Chess is *not* an impartial game because the set of possible moves depends on which player must act next (i.e. only white pieces may be moved on Player 1's turn and only black pieces may be moved on Player 2's turn).

The most famous impartial game is *Nim*, where two players are presented with N piles such that each pile contains $\leq M$ stones. Players alternately remove one or more stones from a single pile at each turn. The player who is able to remove the last stone (leaving their opponent with no stones) wins the game.

We briefly review the known results for Nim here. The analysis of Nim when played with a single pile of stones is trivial. A pile of size k is denoted $*k$, which is called a *nimber*. Player 1 has a winning strategy for a game of Nim with a single pile if and only if $*k \neq *0$.

In order to determine the player who has a winning strategy when Nim is played with more than one pile, the *nim-sum* is computed by taking the exclusive or (XOR) of the pile sizes represented as binary numbers. Player 1 has a winning strategy if and only if the nim-sum of the game is not equal to zero at the beginning of their turn. As an example, consider a game of Nim beginning with three piles containing $3 = 011_2$, $4 = 100_2$, and $5 = 101_2$ stones. The XOR of these values is 010_2 which is equivalent to 2 in base-10, thus Player 1 has a guaranteed winning strategy.

In the above example, the nim-sum of the three piles is denoted as $*3 \oplus *4 \oplus *5 = *2$. This implies that the game of Nim having piles of size 3, 4, and 5 is equivalent to the game of Nim having a single pile of size 2. Because $*2 \neq *0$, Player 1 is guaranteed to have a winning strategy.

Consider a game of Nim with N piles such that each pile contains $\leq M$ stones. By taking the XOR of each pile size represented in binary, we can determine the nimber $*k$ to which the game is equivalent (and thus determine which player has a winning strategy) in $O(N \log M)$ time. Further details of Nim, nimbers, and nim-addition can be found in [1].

According to the *Sprague-Grundy theorem*, an instance of *any* impartial game is equivalent to a nimber $*k$ such that Player 1 has a winning strategy if and only if $*k \neq *0$ [36, 17]. One may wonder whether this leads to an algorithm to solve Kal-Toh-2P$_n(H)$ as follows: construct the instance of Nim that corresponds to it, and then solve it. However, because the proof of this theorem is not constructive in nature, it does *not* immediately give a procedure to determine the nimber corresponding to an instance of Kal-Toh-2P$_n(H)$.

Chapter 16 of [2] gives a detailed analysis of how to find nimbers for instances of the game *dots and boxes*. In this game, players alternately add axis-parallel edges to a grid of points appearing at integer coordinates with the goal of forming 1×1 squares. Note that dots and boxes can be viewed as a simplified version of Kal-Toh-2P$_n(H)$ where H is a 4-cycle and only a limited set of edges can be added. The process of assigning a nimber to an instance of this game is done recursively and requires a non-trivial amount of case analysis (and furthermore, this analysis only holds for grids of a specific size). No simple procedure is given to determine the nimber corresponding to an instance of dots and boxes. It therefore seems even less likely that there is a simple algorithm to find the nimber of Kal-Toh-2P$_n(H)$ for any H.

Instead our analysis treats Kal-Toh-2P$_n(H)$ as an extensive game and uses the idea of a game tree in order to find the player who has a winning strategy.

3.1.2 Kal-toh as an extensive game

The two-player version of Kal-toh is also an *extensive game* with *perfect information*. A simplified version of the definition given in [35] requires the following four conditions for an extensive game:

- A finite set of players.
- A set of all possible sequences of moves such that the final move in each sequence determines the winner. For any two non-identical sequences S_1 and S_2, S_1 cannot be a proper subsequence of S_2.
- Each player has a preference over all of these sequences.
- Given any sequence of moves made so far, it can be immediately determined which player is next to act.

Our formulation of the two-player version of Kal-toh satisfies these four conditions of an extensive game:

- The game is played between two players.
- A sequence of moves is a sequence of edge additions such that the final graph G is the first graph that contains H as a subgraph. No sequence is a proper subsequence of another because the game stops exactly when H becomes a subgraph of G.
- If H is a subgraph of G', then Player 1 wins the game iff G' contains an odd number of edges. This uniquely determines which sequences are preferred by Player 1 and Player 2.
- It is Player 1's turn to act iff G' contains an even number of edges.

The game is said to have perfect information because both players have full knowledge of their opponent's previous moves as well as the set of moves available at any given stage of the game. Assuming that both players act rationally, it is always possible to determine which player has a strategy that guarantees a win. Go and chess (where draws are disallowed) are two examples of other extensive games with perfect information.

3.1.3 The game tree for Kal-toh

For any value of n and H, we can construct a *game tree* representing all possible sequences of edge additions to G terminating when H appears as a subgraph of G. Figure 3.1 shows the game tree for $n = 4$ and $H = K_4$ where the last two levels are omitted (we discuss this below). The construction of the game tree for Kal-Toh-2P$_n(H)$ is always possible because the two-player version of Kal-toh is an extensive game with perfect information. For the remainder of this chapter, we use G to denote the graph corresponding to the current state of the game; initially G is the empty graph.

Consider the game tree for Kal-Toh-2P$_n(H)$ beginning with the empty graph. The player to add an edge alternates at each level of the tree; Player 1 adds an edge to the graphs on odd levels and Player 2 adds an edge to the graph on even levels. We will use the term *safe edge* to denote an edge that may be added to G by a player such that their opponent cannot possibly win on the following turn. On the other hand, adding an *unsafe edge* gives the opponent a winning move on the following turn.

Note that the game is decided once no safe edges can be added. To simplify the game tree, we hence make a leaf of the game tree when no safe edge can possibly be added to the graph. A leaf appearing at an even level implies that Player 1 will win (because Player 2 will be forced to add an unsafe edge) and a leaf appearing at an odd level implies that Player 2 will win (because Player 1 will be forced to add an unsafe edge). Once all leaves have been marked as either P1 (indicating a win for Player 1) or P2 (indicating a win for Player 2), the remaining nodes can be labeled by using the following dynamic programming approach (sometimes called *backwards induction* in a game theory context):

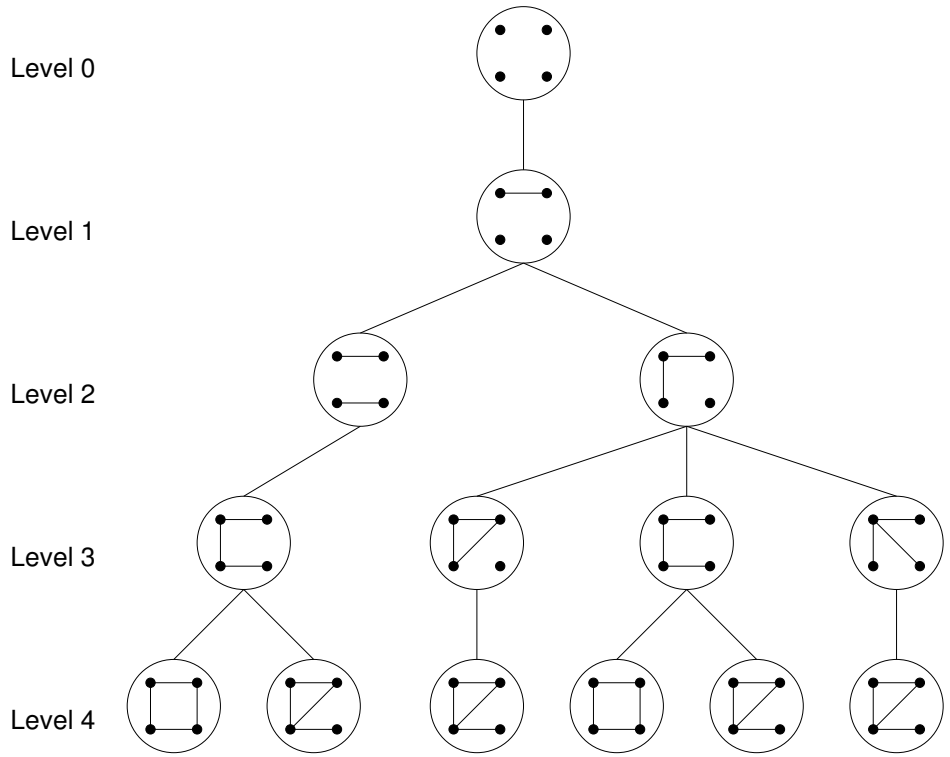

Figure 3.1: The game tree for two-player Kal-toh on $n = 4$ vertices with target subgraph $H = K_4$.

- If it is Player 1's turn to act at node l and at least one child of l appearing at a lower level is labeled as P1, then l is labeled as P1. Player 1 wins at l because there exists a move which leads to a winning configuration for Player 1.
- If it is Player 1's turn to act at node l and all children of l appearing at a lower level are labeled as P2, then l is labeled as P2. Player 1 loses at l because all possible moves lead to winning configurations for Player 2.
- The rules are similar if it is Player 2's turn to act.

It is always possible to construct a game tree and label its nodes in this manner for any extensive game with perfect information [35]. However, the choices made by the winning player may not always be completely unique; in some instances this player may have more than one possible move that will guarantee a win.

So the solution to Kal-Toh-2P$_n(H)$ can be determined by the label attached to the root node of its game tree. Thus the time complexity of solving Kal-Toh-2P$_n(H)$ is determined by the time required to construct its game tree and attach labels to its nodes. By using a directed acyclic digraph instead of a tree, we will show in Section 3.2.1 that a polynomial-time solution to Kal-Toh-2P$_n(H)$ exists when H is a 4-cycle. However, in Section 3.2.2

we provide evidence that solving KAL-TOH-2P$_n(H)$ will likely take exponential time for a general graph H using this approach.

3.1.4 Safe components

We now explore KAL-TOH-2P$_n(H)$ for some very simple graphs H. First consider the case where H is a path of a given length. For all $n \geq 3$, Player 1 always wins if H is a path of length one (i.e. a single edge), and Player 2 will always be able to win if H is a path of length two. If H is a path of length three, then a player can win by either adding an edge between two 1-paths or by adding an edge to an endpoint of a 2-path. Thus Player 1 always has a winning strategy, as Player 2 must either create a second 1-path or a 2-path on his or her first turn.

The analysis becomes more complicated when H is a longer path. For instance, a player could add an edge to a 2-path to form a triangle in order to avoid creating a longer path; such a strategy would be used if the existence of such a path would guarantee a win on the opponent's next turn. The value of n will also have an effect on the game's analysis because the number of vertices places a bound on the number of possible edges and components.

For an arbitrary graph H, this illustration makes it clear that we must approach the analysis of KAL-TOH-2P$_n(H)$ in a more systematic manner. For reasons explained below, we will assume that H contains no bridge and that H does not contain a vertex of degree one. By placing these restrictions on H, we see that the final edge e added to G must be between two vertices belonging to the same component in G. In other words, if H is a subgraph of $G + e$ but is not a subgraph of G, then the graphs $G + e$ and G must have the same number of components.

Define A_H to be the set of components a player must "avoid". The members of this set are all non-isomorphic graphs that can be obtained by removing any one edge from H. If a player were to add an edge e on their turn such that $G + e$ contains any element of A_H as a subgraph, their opponent would have a guaranteed win on the next turn. Because we have made the assumption that H is bridgeless and contains no vertex of degree one, it follows that every graph in A_H must be connected. Thus we have made this restriction to simplify the specification of A_H and the analysis of the game in general.

We can also define S_H, the set of components which are "safe" for a player to create. This is the set of all connected graphs that do not have any graph in A_H as a subgraph. Recall that all graphs in A_H are connected. So if a player adds edge e such that all components of $G + e$ are members of S_H, it is guaranteed that their opponent will not be able to win on the next turn.

Finally, we list all "safe combination rules" for the elements of S_H. These rules describe all possible ways to add an edge to G, either within the same component or between two different components, such that all components of $G + e$ remain in S_H. If a player is unable to follow any of these rules on their turn, then any edge they add will create an element of

A_H as a subgraph of $G+e$, securing a win for their opponent. The only winning strategy for Kal-toh is therefore to trap one's opponent in a situation where no safe rule can be applied.

Given the target subgraph H, a summary of our approach appears below:

1. Determine A_H, the components to avoid. This set contains connected graphs such that one edge can be added to form H.
2. Determine S_H, the safe components. This set contains connected components not containing any of A_H as a subgraph.
3. Determine the set of safe combination rules for the components in S_H.

Consider using this approach to analyze the game KAL-TOH-2P$_n(C_3)$, where C_3 is a 3-cycle:

1. The only element of the set A_H is a 2-path.
2. The set S_H contains two elements: an isolated vertex and a 1-path. These are the only two connected graphs that do not contain a 2-path.
3. The only safe move that can be made using the components in S_H is the addition of an edge between two isolated vertices.

The game will continue as long as it is possible to add an edge between two isolated vertices; the first player forced to add another edge will lose. Thus Player 1 wins KAL-TOH-2P$_n(C_3)$ if $\lfloor \frac{n}{2} \rfloor$ is odd, and Player 2 wins if $\lfloor \frac{n}{2} \rfloor$ is even.

3.2 Strategies

We extend on our analysis of $H = C_3$ by first looking at the game KAL-TOH-2P$_n(C_4)$ in Section 3.2.1. Our approach allows us to determine the winner by constructing the game tree for any n. It is shown in Theorem 5 that it is possible to create the game tree and determine the winner for KAL-TOH-2P$_n(C_4)$ in $O(n^3)$ time.

This approach is then applied to KAL-TOH-2P$_n(H)$ where H is a diamond. In Section 3.2.2 we demonstrate that even less is known about this game as compared to the 4-cycle variant, suggesting that it is likely not possible to find a general solution to KAL-TOH-2P$_n(H)$ where $H = K_4$ (or H is a more complicated planar and triconnected graph) by examining its game tree; a 4-cycle and a diamond were selected to study initially because they are a few edges away from K_4.

3.2.1 Kal-toh on a 4-cycle

Safe components and rules for 4-cycles

Let H be a 4-cycle. In order to avoid losing the game, a player would not want to create any component containing a 3-path, otherwise their opponent will be able to win on the next turn. Removing a single edge from a 4-cycle will always result in a 3-path, thus the only element of the set A_H is a 3-path.

From this we can list the graphs in S_H: a singleton vertex, a single edge, a 2-path, a triangle, and $K_{1,t}$ where $t \geq 3$ (i.e. a star on at least four vertices). To see that there are no other graphs in the set S_H, consider the maximum degree of such graphs $S \in S_H$. If S contains a vertex v of degree at least three and is not a star, then there exists an edge not incident to v and therefore S must contain a 3-path. So S is either a star, or the maximum degree of S must be two. The 3-path is on four vertices, two of which have degree two. If S were to have at least two vertices of degree two, it must be on fewer than four vertices; the only graph to satisfy these conditions is the triangle. Finally, the only possible connected graph with exactly one degree-two vertex is the 2-path, and the only possible connected graphs with no degree-two vertices are the 1-path and singleton vertex.

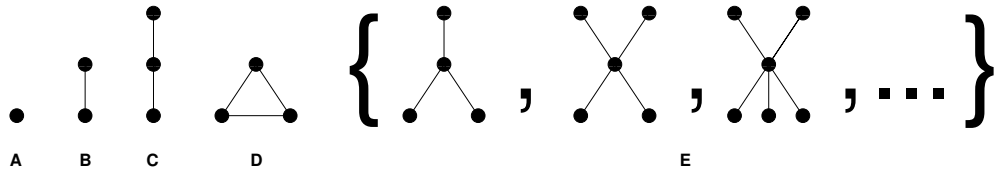

Figure 3.2: The elements of S_H, the safe components when H is a 4-cycle.

Each of these safe components are assigned a label $A, \ldots, E$ as shown in Figure 3.2. Note that components A, B, C, and D are fixed graphs, while E represents the set of all stars on at least four vertices. Using these labels we define the following set of safe combination rules:

1. A, $A \to B$
2. A, $B \to C$
3. $C \to D$
4. A, $C \to E$
5. A, $E \to E$

For instance, the rule A, $A \to B$ says that an edge may be safely added between two isolated vertices (A, A) to create a component containing a single edge (B). The rule $C \to$

D says that one edge may be safely added to the endpoints of a two-path (C) to form a triangle (D).

In order to show that these are the only possible combination rules, we work backwards and consider the effect of removing a single edge e from the components B, C, D, and E. The graph $B - e$ consists of two isolated vertices (A, A), the graph $C - e$ consists of an isolated vertex and a single edge (i.e., A and B), and the graph $D - e$ is a 2-path (i.e., C). If E is a star on four vertices, then the graph $E - e$ consists of an isolated vertex and a 2-path (A and C). Finally, if E is a star on $t > 4$ vertices, then the graph $E - e$ consists of an isolated vertex and a star on $t - 1$ vertices (A and E).

Recall that we only have the safe components A, B, C, D, and E in any configuration of the game tree. To simplify the notation, we will represent the configuration as a string of the form $A^{n_A}B^{n_B}C^{n_C}D^{n_D}E^{n_E}$ where each exponent represents the number of safe components of that type currently present in the graph. Symbols having an exponent of zero will be omitted. For example, the string A^3B^2D represents the graph having three isolated vertices, two single edges, and one triangle (six components in total).

We will call a configuration *terminal* if no safe rules can be applied. In this case, G contains no C component and if it contains an A component then it contains neither a B nor an E component, nor another A component.

A sequence of edge additions to G can be represented by a series of combination rules applied to the current configuration of the graph. As an example, consider the graph A^4. One possible sequence of combination rules is $A^4 \xrightarrow{(1)} A^2B \xrightarrow{(2)} AC \xrightarrow{(3)} AD$, where the rule applied is shown in parentheses; see Figure 3.3. We will call such a sequence of rules $A^4 \rightsquigarrow AD$ a **derivation**.

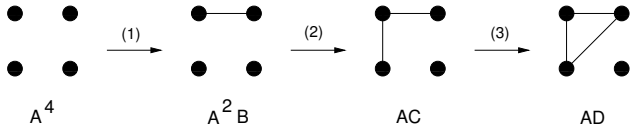

Figure 3.3: A derivation from A^4 to AD using the rules listed in parentheses.

Starting with some configuration of G, we can show all possible derivations in a game tree and apply the analysis previously described, at which point the root of the game tree will indicate the player who has a strategy that guarantees a win. Therefore, the winner of any game of Kal-Toh where H is a 4-cycle can be determined by constructing its game tree. The example in Figure 3.4 shows the game tree with the starting configuration A^6, the graph with six vertices and no edges. Note that we will eventually combine repeated configurations by using a directed acyclic graph (DAG) instead of a game tree (the DAG for A^7 is given in Figure 3.5).

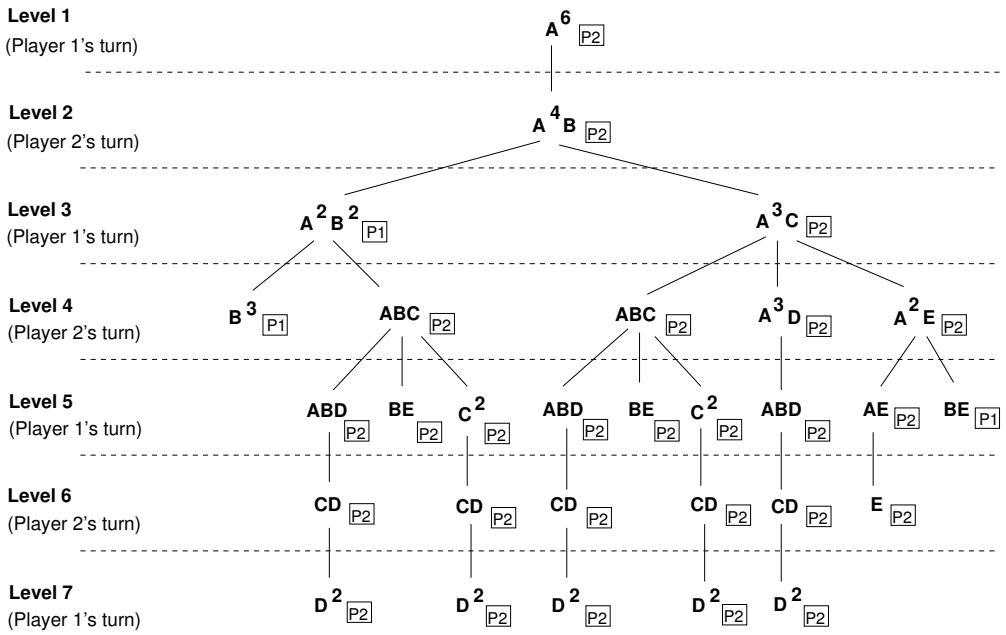

Figure 3.4: The game tree rooted at A^6 for Kal-Toh-2P$_6(C_4)$.

The height of the game tree

If all leaves of the game tree for Kal-Toh-2P$_n(C_4)$ (for some fixed value of n) appeared only at even-numbered levels or only at odd-numbered levels, then we would easily be able to determine the winner of the game. However, the example seen in Figure 3.4 shows that this is not always the case.

The following lemma gives the height of the game tree for Kal-Toh-2P$_n(C_4)$. While this result does not seem to provide any additional information about the winner of the game, it does give an upper bound on the number of edges that can be added to G before the addition of an unsafe edge is forced.

Lemma 14. *Let T be the game tree for* Kal-Toh-2P$_n(C_4)$ *where $n \geq 4$. Then the height of T is n (if $n \equiv 0 \pmod 3$) and $n-1$ otherwise.*

Proof. We first make the observation that there exist derivations from A^3 and A^4 to a terminal configuration in three steps: $A^3 \xrightarrow{(1)} AB \xrightarrow{(2)} C \xrightarrow{(3)} D$ and $A^4 \xrightarrow{(1)} A^2B \xrightarrow{(2)} AC \xrightarrow{(4)} E$. Furthermore, there exists a derivation from A^5 to a terminal configuration in four steps: $A^5 \xrightarrow{(1)} A^3B \xrightarrow{(2)} A^2C \xrightarrow{(4)} AE \xrightarrow{(5)} E$.

Using these observations, we show a lower bound on the height of T in the following three cases.

Case 1 ($n = 3k$). The game beginning with $A^n = A^{3k} = (A^3)^k$ has a derivation $A^{3k} \rightsquigarrow D^k$. Each derivation of $A^3 \rightsquigarrow D$ uses three steps, hence $A^{3k} \rightsquigarrow D^k$ uses $3k$ steps. Therefore the height of T is at least $3k = n$.

Case 2 ($n = 3k + 1$). Consider the game $A^n = A^{3k+1} = A^4 \cdot A^{(3k+1)-4} = A^4 \cdot (A^3)^{k-1}$. The derivation $A^4 \rightsquigarrow E$ uses three steps and the derivation $(A^3)^{k-1} \rightsquigarrow D^{k-1}$ uses $3(k-1)$ steps. Therefore the height of T is at least $3 + 3(k-1) = 3k = n - 1$.

Case 3 ($n = 3k+2$). Consider the game $A^n = A^{3k+2} = A \cdot A^4 \cdot A^{(3k+2)-5} = A \cdot A^4 \cdot (A^3)^{k-1}$. The derivation $A^5 \rightsquigarrow E$ uses four steps, and the derivation of $(A^3)^{k-1} \rightsquigarrow D^{k-1}$ uses $3(k-1)$ steps. Therefore the height of T is at least $4 + 3(k-1) = 3k + 1 = n - 1$.

We now argue that these values also act as an upper bound on the number of edges in the graph G, which is equivalent to the height of T. Let G be a graph on n vertices with m edges such that no further derivation rule can be applied. Thus, G is the graph at a leaf of the game tree, does not contain any 3-path, and any component of G is A, B, C, D, or E. Because G is terminal, it is either AD^{n_D} or $B^{n_B}D^{n_D}E^{n_E}$ (where $n_B, n_D, n_E = 0$ is possible).

Any component of G with l vertices has $l - 1$ edges except for D, which has l edges. If $n = 3k$, then m is at a maximum if G consists of k triangles (D components), in which case $m = n$. In all other cases, G has at least one component that is not D, and hence has at most $n - 1$ edges.

□

Corollary 4 is the contrapositive of Lemma 14, which is of possible independent interest.

Corollary 4. *Let G be a graph on n vertices and m edges. If $m > n$ when $n \equiv 0 \pmod 3$ or $m > n - 1$ otherwise, then it is possible to add an edge e to G such that $G + e$ contains a 4-cycle.*

Solving Kal-Toh-2P$_n(C_4)$

Recall that our ultimate goal is to determine the winner of Kal-Toh-2P$_n(C_4)$ beginning with an arbitrary graph G in $O(1)$ time without needing to construct the game tree. Doing so would require us to find some sort of pattern based only on the number of each component present in the initial game configuration. Lemma 15 gives an example of a $O(1)$-time method to solve a simple instance of the game.

Lemma 15. *In the game* Kal-Toh-2P$_n(C_4)$, *$n \geq 4$, with $G = C^{n_C}B^*D^*E^*$, Player 1 wins if n_C is odd and Player 2 wins if n_C is even.*

Proof. The only rule that can be applied is $C \xrightarrow{(3)} D$. Therefore, the only possible derivation is $B^*C^{n_C}D^{n_D}E^* \rightsquigarrow B^*C^0D^{n_D+n_C}E^*$ via a sequence of n_C applications of Rule 3. The terminal configuration appears at level $n_C + 1$ of the game tree, so Player 1 wins if n_C is odd and Player 2 wins if n_C is even. □

An $O(1)$-time solution to the general problem unfortunately remains unknown for the general game, but we can show that it is possible to obtain an answer in polynomial time.

Theorem 8. *For any configuration $A^{n_A}B^{n_B}C^{n_C}D^{n_D}E^{n_E}$ having n vertices total, we can determine in $O(n^3)$ time which player can be the first to create a 4-cycle.*

Proof. Consider the game tree for KAL-TOH-2P$_n(C_4)$ beginning with the arbitrary configuration $A^{n_A}B^{n_B}C^{n_C}D^{n_D}E^{n_E}$. Because there are at most five rules that can be applied to any configuration, and because the game tree has height at most n by Lemma 14, the game tree will have $O(5^n)$ nodes in the worst case. This means that determining the winner of the game by constructing the game tree could possibly take exponential time.

In order to reduce the space needed to represent the possible game configurations, we use a directed acyclic digraph (DAG) having unique nodes instead of a game tree; the DAG corresponding to the game A^7 is shown in Figure 3.5. Any configuration has $n_A \leq n$, $n_B \leq \frac{n}{2}$, and $n_C \leq \frac{n}{3}$. Furthermore, the value of n_D is irrelevant because D components do not appear on the left-hand side of any derivation rule; we therefore contract these configurations into a single node where the value of n_D can be anything. The exact value of n_E is irrelevant because we only need to know if the value of n_E is either 0 or at least one, so we apply a similar contraction to these configurations. In total, there are at most $n \cdot \frac{n}{2} \cdot \frac{n}{3} \cdot 2 \in O(n^3)$ configurations of interest in the DAG, where we group together nodes having the same configuration. Thus the winner of the game can be determined in $O(n^3)$ time by using the dynamic programming approach explained below.

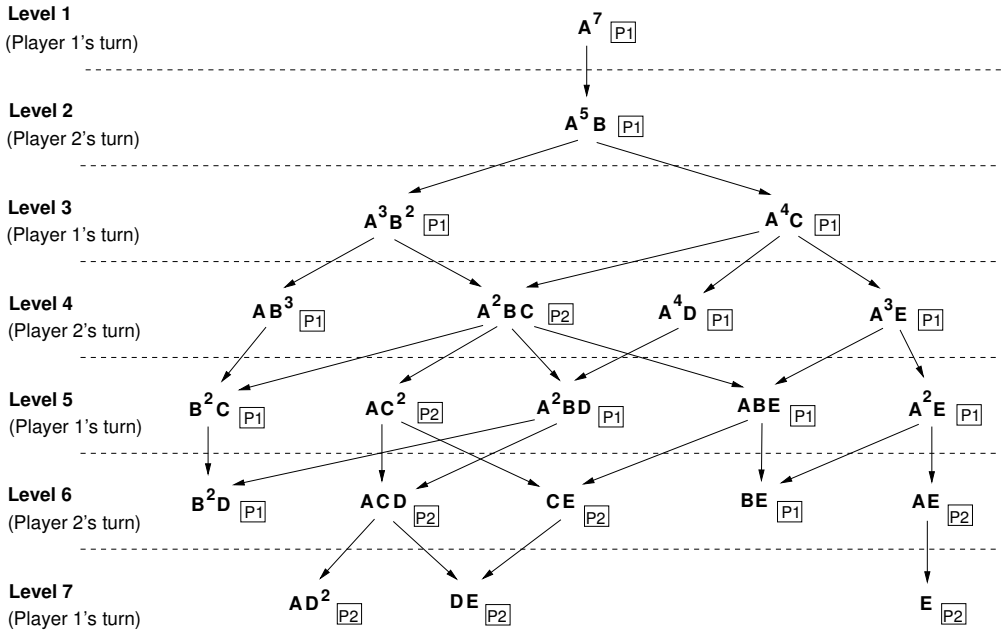

Figure 3.5: The DAG rooted at A^7 for KAL-TOH-2P$_7(C_4)$.

Let S be a table indexed by all configurations where $n_A \leq n$, $n_B \leq \frac{n}{2}$, $n_C \leq \frac{n}{3}$, and where $n_E = 0$ (there are no components of type E) or $n_E = 1$ (there is at least one component of type E). We would like to set the value of $S[i]$ to be 0 if the current player will lose or 1 if the current player will win. This means that Player 1 has a winning strategy if and only if $S[i_0] = 1$, where i_0 is the configuration $A^{n_A}B^{n_B}C^{n_C}D^{n_D}E^{n_E}$.

To fill the entries of S, we first assign a value of 0 to each entry corresponding to terminal configurations, i.e., configurations of the form $A^0B^*C^0D^*E^*$ and $A^1B^0C^0D^*E^0$. The remaining entries in S are filled by recursively applying backwards induction ending with configuration i_0. The time to compute each $S[i]$ is $O(1)$ because there are only five rules, thus each entry has at most five outgoing arrows in the corresponding DAG. Because the value $S[i]$ is computed once for each of the $O(n^3)$ possible configurations i, it will take $O(n^3)$ time to fill all of the entries of S.

□

Corollary 5. KAL-TOH-2P$_n(C_4)$ *can be solved in* $O(n^3)$ *time.*

Improvements to this bound may be possible. In light of Lemma 15, we suspect that it may be sufficient for n_C to only represent "even or odd" instead of the actual number of components of type C. This simplification would reduce the running time to $O(n^2)$, but would fall short of the ultimate goal of an $O(1)$-time rule to determine which player has a winning strategy.

3.2.2 Larger Forbidden Subgraphs

In Section 3.2.1 we looked at KAL-TOH-2P$_n(H)$ where H was a 4-cycle. This was just a first step towards the ultimate goal: solve this game where H is triconnected and planar, thus representing a polyhedron.

Moving one step closer, we consider KAL-TOH-2P$_n(H)$ where H is a diamond, the graph obtained by removing a single edge from K_4. There are two elements in the set A_H, the graphs that are one edge away from H: the 4-cycle and the triangle with one extra edge. See Figure 3.6.

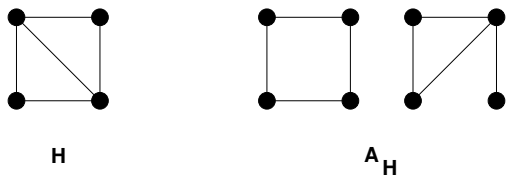

Figure 3.6: A_H is the set of graphs that can create a diamond, H, by the addition of one edge.

Recall that we were easily able to give a list of five safe component types, S_H, when H was a 4-cycle: a singleton vertex, an edge, a 2-path, a triangle, and any star on at least four vertices. Although the last component type represented a family of infinitely many graphs (i.e. stars on $n \geq 4$ vertices), we were still able to give an exact description of how an edge could be added safely between a star and another component type. Namely, an edge may be added between a 2-path and an isolated vertex to form a star on four vertices (Rule 4), or an edge may be added between an isolated vertex and a star on k vertices to form a star on $k+1$ vertices (Rule 5). The degree of any star's central vertex was irrelevant.

Now consider the set of safe components S_H when H is a diamond. A connected graph is an element of this set if it does not contain any graph in A_H as a subgraph. Thus three categories of graphs will belong to S_H:

- Any tree
- A triangle
- Any graph on at least four vertices with girth (i.e., the length of the shortest cycle) at least five

No other graphs will be in S_H because they will contain a 3-cycle with attached edge or a 4-cycle. These categories include a very wide range of graphs, making it difficult to list precisely all of the safe combination rules. For instance, it is not always possible to add an edge between two vertices of a tree such that the resulting graph has girth five. Similarly, it may not be possible to add an edge that maintains a girth of five to a graph already having a girth of five. Figure 3.7 shows examples of these cases.

It is possible to store extra information about each component to determine if the addition of a safe edge is possible. For example, the length and endpoints of a longest path of a tree could be maintained; if this length is at least four, then one edge could be added to form a graph having girth at least five. However, these values would need to be re-computed each time an edge is added to join two trees. This prevents us from doing a simple dynamic programming exploration as we did for $H = C_4$. It remains open whether Kal-Toh-2P$_n(H)$ can be solved in polynomial time when H is a diamond.

Now consider Kal-Toh-2P$_n(H)$ for an an arbitrary triconnected planar graph H. Computing A_H is easy, as there are at most $m(H)$ graphs in this set. However, even *describing* S_H is nontrivial; this involves checking whether any element of A_H appears as a subgraph of G, which is equivalent to the SubgraphIsomorphism problem. This problem is NP-complete if G is non-planar [11].

If we wanted to give a precise description of the graphs H for which Kal-Toh-2P$_n(H)$ had a polynomial-time solution, then the following generalization of Theorem 5 would be a worthwhile direction to investigate: Kal-Toh-2P$_n(H)$ can be solved in polynomial time if S_H can be "nicely" described by finitely many sets of components. Note that an exact

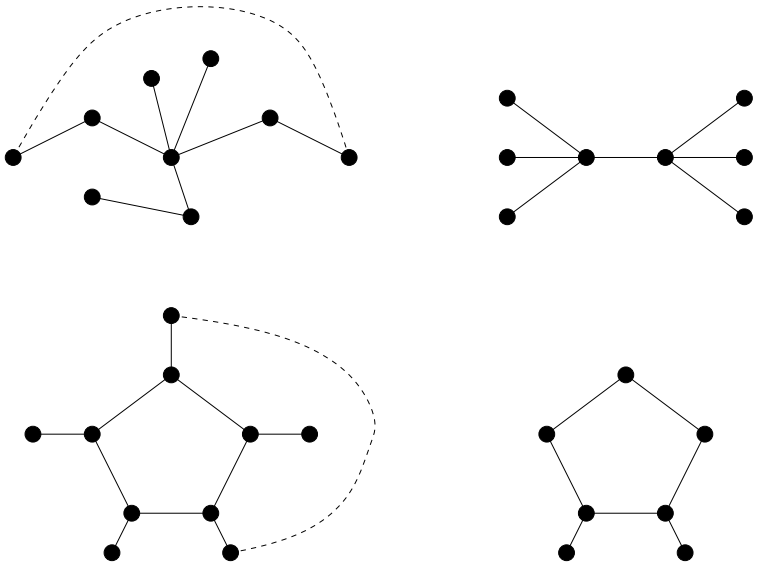

Figure 3.7: The dashed edge can be added to the tree in the top-left to form a graph with girth five, whereas no edge addition is possible to the tree in the top-right. The dashed edge can be added to the graph in the bottom-left that maintains a girth of five, whereas no edge addition is possible to the graph in the bottom-right.

description of "nicely" is unclear; the set S_H can be "nicely" described when H is a 4-cycle, but (probably) not when H is a diamond. A nice description seems to require a finite, deterministic list of combination rules that can be blindly applied to the component types within S_H without needing to consider any additional properties of these graphs. Computing safe combination rules appear just as difficult, hence it is reasonable to conjecture that deciding Kal-Toh-2P$_n(H)$ is NP-hard for sufficiently complicated H, though a formal proof of this remains to be done.

Chapter 4

Conclusions

The topics selected for this thesis were originally motivated by the fictional game of *Kal-toh* seen on the television series *Star Trek: Voyager*. Kal-toh is played with a jumbled collection of small metal rods, with the ultimate goal of forming a polyhedron by using a subset of these rods as its edges. The game is depicted as one that could either be played alone or against an opponent. In either case, a single turn consists of adding a rod somewhere within the existing structure.

Because the exact rules of the game were never formally explained to the viewer, it was necessary to first give a more precise description based on our own interpretation of the game. By having an edge represent a single rod and having a vertex represent the point of contact between two rods, we formulated the single-player and the two-player versions of the game.

Chapter 2 focused on the single-player version of Kal-toh. Our formulation of the decision problem KAL-TOH$_{l,\geq k}$ asked if it is possible to create a triconnected planar subgraph on at least k vertices with the addition of at most l edges. We first considered instances of the problem where no edges were to be added to the graph (i.e. when $l = 0$). Using a reduction from the NONCROSSINGCYCLE problem, we showed that KAL-TOH$_{0,\geq 4}$ is NP-hard and gave a similar proof of the NP-hardness of KAL-TOH$_{0,\geq \frac{3n}{4}}$.

We suspect that KAL-TOH$_{0,\geq k}$ is NP-hard even for $k = n$ but have not been able to provide a proof of this claim. If this were false, then there would exist an efficient way to remove edges from a non-planar triconnected graph G until planarity is achieved such that the removal of these edges does not violate the triconnectivity of G. This would suggest the additional open problem, "what is the smallest value of c for which KAL-TOH$_{0,\geq n-c}$ is NP-hard?"

Section 2.2 considered instances of KAL-TOH$_{l,=k}$ for which edge additions to the graph are permitted. We started by showing a few results for small values of l and conjectured that KAL-TOH$_{2,\geq 4}$ and KAL-TOH$_{1,\geq 4}$ are both NP-hard, but have not been able to prove either of these. By specifically creating a prism as our triconnected planar subgraph, we

demonstrated one upper bound for l. A second approach gives an upper bound of $l \leq k+1$ for connected graphs, which means that the answer to KAL-TOH$_{k+1,=k}(G)$ is YES when G is connected. However, showing that this bound is tight for connected graphs in general remains open. Finally, we prove that a graph is triconnected if it contains a Schnyder wood (along with three additional edges), which may be of independent interest.

Chapter 3 considered the two-player version of Kal-toh. We formulated this problem as KAL-TOH-2P$_n(H)$, in which players alternately add edges to the graph; the first player to create an instance of H as a subgraph of G is declared the winner. Our goal was to determine if the general version of this game can be solved in polynomial time, with $O(1)$ time being ideal. We showed that the specific instance KAL-TOH-2P$_n(C_4)$ can be solved in $O(n^3)$ time and conjectured that there exists no polynomial-time solution to KAL-TOH-2P$_n(H)$ in general.

Alternate interpretations of Kal-toh suggest other directions for future study. A more restrictive interpretation of the single-player version could require H to be an *induced* triconnected planar subgraph of G. If the induced version of KAL-TOH$_{l,\geq k}$ gives an answer of YES, then the original formulation of KAL-TOH$_{l,\geq k}$ must also have an answer of YES because every induced subgraph is a subgraph of G. The opposite direction is not necessarily true (with $K_{4,4}$ being a counterexample), but even the induced version of KAL-TOH$_{0,\geq k}$ is NP-hard.

However, some of our results do not continue to hold when the triconnected planar subgraph H must be an induced subgraph of G. For instance, it is not true that any connected graph on k vertices will have a triconnected planar induced subgraph after the addition of at most k edges based on our proof of this claim for the non-induced case. Our technique added at most k edges to a spanning tree on k vertices such that planarity was maintained with each edge addition. This does not always work for the induced case because when the initial k vertices are selected, the corresponding induced subgraph is not necessarily a tree, and so our procedure may introduce forced crossings as edges are added.

Another possible direction of study could focus on the interpretation of Kal-toh as a geometric graph in three dimensions. If the vertices of G are at fixed coordinates, one might be interested in determining if it is possible to form a triconnected planar subgraph given a set of m rods having fixed lengths $l_1, l_2, \ldots, l_m$. A similar problem is studied in [32]. Along these same lines, we could also assign a thickness of $t > 0$ to each rod and impose the restriction that no two rods may intersect. Similar questions could be asked in the cases where vertices are allowed to move or where rod lengths were not prescribed before being added to the game.

It should be mentioned that the original depiction of Kal-toh in *Star Trek: Voyager* shows players moving an edge from one part of the game configuration to another on each turn, instead of adding a new edge as specified in our formulation, which suggests another possible variant of the game that could be studied.

All of the problems we have studied are special cases of the more general question: Is it possible for a graph G with properties P_G to contain a subgraph H with properties P_H

after the application of at most ℓ operations from the set Q? Future work could focus on the solution of this problem for many different values of P_G, P_H, ℓ, and Q.

References

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*, volume 1. Academic Press, 1982.

[2] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*, volume 2. Academic Press, 1982.

[3] Nicolas Bonichon, Stefan Felsner, and Mohamed Mosbah. Convex Drawings of 3-Connected Plane Graphs. In *Graph Drawing*, Lecture Notes in Computer Science, pages 60–70. Springer, 2005.

[4] F. R. K. Chung. Separator theorems and their applications. In B. Korte, L. Lovasz, H. J. Promel, and A. Schriver, editors, *Paths, Flows, and VLSI-Layout*, pages 17–34. Springer-Verlag, 1990.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill, second edition, 2001.

[6] Gruia Călinescu, Cristina G. Fernandes, Ulrich Finkler, and Howard Karloff. A better approximation algorithm for finding planar subgraphs. In *SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 16–25, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[7] G. Di Battista and R. Tamassia. Incremental planarity testing. In *30th Annual Symposium on Foundations of Computer Science*, pages 436–441, 1989.

[8] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, April 1996.

[9] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, third edition, 2006.

[10] Frederic Dorn. Planar Subgraph Isomorphism Revisited. In *27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, volume 5, pages 263–274, September 2010.

[11] David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 632–640, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[12] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.

[13] Stefan Felsner. Convex Drawings of Planar Graphs and the Order Dimension of 3-Polytopes. *Order*, 18(1):19–37, March 2001.

[14] Zvi Galil, Giuseppe F. Italiano, and Neil Sarnak. Fully dynamic planarity testing with applications. *J. ACM*, 46(1):28–91, 1999.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W.H. Freeman & Company, first edition, January 1979.

[16] A. J. Goldstein. An Efficient and Constructive Algorithm for Testing Whether a Graph Can Be Embedded in a Plane. In *Graph and Combinatorics Conference*. Office of Naval Research Logistics Project, Department of Mathematics, Princeton University, 1963.

[17] P. M. Grundy. Mathematics and Games. *Eureka*, 2:6–8, 1939.

[18] Carsten Gutwenger, Michael Jünger, Sebastian Leipert, Petra Mutzel, Merijam Percan, and René Weiskircher. Subgraph Induced Planar Connectivity Augmentation. In *Workshop on Graph Theoretic Concepts in Computer Science (WG'03)*, volume 2880 of *Lecture Notes in Computer Science*, pages 261–272. 2003.

[19] Carsten Gutwenger and Petra Mutzel. A Linear Time Implementation of SPQR-Trees. In *Graph Drawing (GD'00)*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. 2001.

[20] Carsten Gutwenger, Petra Mutzel, and Bernd Zey. On the Hardness and Approximability of Planar Biconnectivity Augmentation. In *The 15th International Computing and Combinatorics Conference (COCOON'09)*, volume 5609 of *Lecture Notes in Computer Science*, pages 249–257. 2009.

[21] B. Haeupler and R. Tarjan. Planarity Algorithms via PQ-Trees (Extended Abstract). *Electronic Notes in Discrete Mathematics*, 31:143–149, August 2008.

[22] J. E. Hopcroft and R. E. Tarjan. Dividing a Graph into Triconnected Components. *SIAM Journal on Computing*, 2(3):135–158, 1973.

[23] John Hopcroft and Robert Tarjan. Efficient Planarity Testing. *J. ACM*, 21(4):549–568, October 1974.

[24] T. Hsu. On Four-Connecting a Triconnected Graph. *Journal of Algorithms*, 35(2):202–234, May 2000.

[25] T. S. Hsu and V. Ramachandran. A linear time algorithm for triconnectivity augmentation. *Symposium on Foundations of Computer Science*, pages 548–559, 1991.

[26] Tsan-sheng Hsu and Vijaya Ramachandran. On Finding a Smallest Augmentation to Biconnect a Graph. *SIAM Journal on Computing*, 22, 1993.

[27] Michael Jünger and Petra Mutzel. The polyhedral approach to the maximum planar subgraph problem: New chances for related problems. In *Graph Drawing*, Lecture Notes in Computer Science, pages 119–130. Springer, 1995.

[28] Goos Kant. *Algorithms for drawing planar graphs.* PhD thesis, Utrecht University, Utrecht, Netherlands, June 1993.

[29] Goos Kant and Hans Bodlaender. Planar graph augmentation problems. In *Proceedings of the 2nd Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, pages 286–298. Springer, 1991.

[30] Jan Kratochvil, Anna Lubiw, and Jaroslav Nešetřil. Noncrossing Subgraphs in Topological Layouts. *SIAM Journal on Discrete Mathematics*, 4(2):223–244, 1991.

[31] P. C. Liu and R. C. Geldmacher. On the deletion of nonplanar edges of a graph. In *Proc. of the 10th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 727–738, 1979.

[32] Brendan Lucier. Local overlaps in special unfoldings of convex polyhedra. *Comput. Geom. Theory Appl.*, 42:495–504, July 2009.

[33] Kurt Mehlhorn, Petra Mutzel, and Stefan Näher. An Implementation of the Hopcroft and Tarjan Planarity Test and Embedding Algorithm. Technical report, Zentrum für Angewandte Informatik Köln, Lehrstuhl Jünger, 1993.

[34] Petra Mutzel. A polyhedral approach to planar augmentation and related problems. In *Third Annual European Symposium on Algorithms*, pages 494–507, 1995.

[35] M. J. Osborne. *An Introduction to Game Theory.* Oxford University Press, New York, 2004.

[36] R. P. Sprague. Über mathematische Kampfspiele. *Tohoku Mathematical Journal*, 41:438–444, 1936.

[37] Maciej Sysło and Andrzej Proskurowski. On Halin graphs. In M. Borowiecki, John W. Kennedy, and Maciej M. Sysło, editors, *Graph Theory*, volume 1018 of *Lecture Notes in Mathematics*, chapter 31, pages 248–256. Springer, 1983.

[38] Robert Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[39] Toshimasaa Watanabe, Tadashia Ae, and Akiraa Nakamura. On the NP-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, May 1983.

[40] Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264, New York, NY, USA, 1978. ACM.

[41] Mihalis Yannakakis. The Effect of a Connectivity Requirement on the Complexity of Maximum Subgraph Problems. *J. ACM*, 26(4):618–630, 1979.

[42] Günter M. Ziegler. *Lectures on Polytopes*, chapter 4, pages 104–126. Graduate Texts in Mathematics. Springer-Verlag, 1995.