Feedback Vertex Kayles

Finding a wining strategy in a two-player combinatorial game

Geertièn de Vries

ICA-3588165
Utrecht University, The Netherlands

August 4, 2016

Abstract

In this thesis Feedback Vertex Kayles is discused. Feedback Vertex Kayles is a two-player combinatorial game on graphs in which players remove nodes positioned on cycles. We also use the name Feedback Vertex Kayles to denote the problem of exploring whether there is a winning strategy for the first player and most often what this strategy is. This problem is an interesting and complex problem expected to be PSPACE-complete. Apart from discussing the complexity of the problem as a whole, in this thesis we will look into Feedback Vertex Kayles on specific graph classes. We discuss a number of graph classes for which we can determine in polynomial or faster time whether there is a winning strategy. To help the reader better understand the specific results and proofs and to give a deeper understanding of Feedback Vertex Kayles, this thesis first gives a broad range of important definitions and basic insights for Feedback Vertex Kayles.

1 Introduction

Combinatorial games are an interesting field of study. They are interesting not so much for (direct) relations to practical problems, although those do exist, but more because of the mathematical complexity of most of these games. That they are a good field for studies is reinforced by the large amount of literature written about combinatorial games. [Fraenkel, 1996] gives a vast bibliography on the subject along with a more extensive argument on what makes combinatorial games so interesting.

Many combinatorial games are found to be PSPACE-complete. One of the first to prove the mathematical complexity of a number of combinatorial games was Schaefer [Schaefer, 1976]; [Schaefer, 1978]. One of the interesting games he introduced and proved to be PSPACE-complete in this paper is Node Kayles. In more recent literature Node Kayles is mostly simply refered to as Kayles; in the sequel we will do so as well. It should be noted however that Kayles is also the name of a much older game on which Node Kayles is based. In this original game there is a row of pins and two players in turn throw a ball at the pins. The idealized version of this game has been studied by mathematicians. In this version the assumption is that a player can always throw over one pin or two directly adjacent pins. The player can never throw over less than one or more than two pins and can never throw over two pins that are spaced further apart. In a general game with one row of adjacent pins the first player will win the game, but more can be set about the values of different positions as first thoroughly addressed by [Guy and Smith, 1956]. We will look further into this game later, as it has a strong connection to the game we analyze.

Although Kayles has been proven to be PSPACE-complete, research does not stop there. There are several interesting directions for research. Most interesting are fast (polynomial time) exact algorithms for special cases and a fast (but exponential time) exact algorithm for general graphs.

Research results in both directions depend on the use of nimbers [Berlekamp et al., 1982]. These nimbers are game values related to stacks in the game of Nim. As explained clearly by [Bodlaender and Kratsch, 2002], Kayles is a game adhering to some specifications that make it possible to assign a nimber to every position in the game. Their use of nimbers for Kayles is based on Sprague-Grundy theory. Nimbers and Sprague-Grundy theory are further discussed in Section 2.2. Good books going further into the theory are written by [Conway, 1976] and [Berlekamp et al., 1982].

It is possible to look at fast exact algorithms to solve Kayles in a general case. However, these algorithms will remain exponential. A fast exact exponential time algorithm using nimbers is described by [Bodlaender et al., 2015].

Some special cases can be solved in polynomial time. These cases have special

graph structures on which a certain algorithm different from the standard algorithm can determine quicker whether there is a winning strategy or not. One good example of a paper that does this for several graph structures is by [Bodlaender and Kratsch, 2002].

Apart from research into Kayles itself, variations on Kayles are an interesting research area. One variation that has been an open problem for a long time is Arc Kayles. The problem was already proposed by [Schaefer, 1978]. In this game a player chooses an edge and removes that edge and adjacent edges, as opposed to nodes in Node Kayles. To this day the complexity of the problem whether the first player has a winning strategy in this game has not been determined. Some research has been done on the game though. A recent example is by [Huggan, 2015], who gives a number of fast algorithms for special cases. She also conjectures that the problem is in P. Another interesting variation is discussed by [Prins, 2015]. The problem he discusses is Kayles with marking or removal on arbitrary distance, as opposed to distance one. He proves this problem to be PSPACE-complete and gives some exact polynomial time algorithms for special cases.

This thesis also studies a variation on Kayles. The variation gives a rather different game that we call Feedback Vertex Kayles. In this game both players alternately choose a vertex in the graph that is on at least one cycle and remove it and any incident edges from the graph. A move can cause nodes to no longer be on any cycle. Nodes that are not on any cycle cannot be chosen for a later move, and thus for the sake of clarity can be removed as well. The game ends when no more nodes (that are on at least one cycle) remain. The player that is to play at that moment loses the game. In other words the player to make the last move wins the game. The name and idea come from combining Kayles with the Feedback Vertex Set problem. In the Feedback Vertex Set problem a minimum size set of vertices has to be found such that when all these vertices are removed from the graph, all cycles are removed from the graph. This problem has been proven to be NP-complete. Note that a good strategy for playing Feedback Vertex Kayles does not imply playing on a minimal Feedback Vertex Set.

2 Definitions

In this section some important definitions for the rest of the thesis are given.

2.1 Feedback Vertex Kayles

Let us first define Feedback Vertex Kayles more clearly. Feedback Vertex Kayles is a two-player game played on a simple undirected graph. In turn each player chooses a node in the graph that lays on at least one simple cycle in the graph

and removes it from the graph. Any node that is not on a cycle is not a node to remove as a legal move. To make the situation in a game more clear, nodes that are not on any cycle can safely be removed from the game as this does not remove or add cycles to the game and as mentioned these nodes cannot be used for a move. Removing these nodes does not change the game. When evaluating the game with an automated algorithm, it is very important that the algorithm does not wrongly consider nodes that are not on a cycle as possibilities for moves. Considering these nodes as candidates for moves will change the game and possibly whether there is a wining strategy for the first player. The game ends when there are no more nodes in the graph that are on at least one cycle and thus there are no more legal moves. The player that is to move when the game ends loses the game. In other words the player that made the last move wins.

The game gives an interesting problem to solve: the question whether or not there is a wining strategy for the first player to move on a certain graph. Note that the game can only be won by one player and lost by the other, there are no draws or other results. There is a wining strategy for the first player if for a certain first move, for every later move that the second player can make in the resulting situation there is a move the first player can make so that finally the last move is made by the first player. In the sequel we will refer to this problem as Feedback Vertex Kayles as well. It should be clear from context whether by Feedback Vertex Kayles the game or the problem is meant at all times.

2.2 Sprague-Grundy theory

Important for the analyses of many combinatorial games is Sprague-Grundy theory. This theory was invented individually by both [Sprague, 1935] and [Grundy, 1939] in the nineteen thirties. In this subsection we will shortly discuss the important parts of the theory and how we can apply them to Feedback Vertex Kayles. For a deeper but non-formal insight into Sprague-Grundy theory and some (fun) applications of the theory the reader is referred to [Berlekamp et al., 1982]. For a more formal deeper insight the reader is referred to [Conway, 1976].

In the theory integer values are assigned to positions in a game. These values are based on positions in the game of Nim. They are therefore also often called nimbers, as done by [Berlekamp et al., 1982]. In the sequal of this thesis we will also call these integers nimbers. More formally a *nimber* is an integer $i \in N_0 = 0, 1, 2, ...$ When we have a specific position p in a game we denote the nimber for that position $nb(p)$.

Sprague Grundy theory is applicable to each combinatorial game that adheres to the following specifications. It

- is a two player game.

- has a "last player to move wins"-rule. In other words when there are no more possible moves in a certain position the player to move loses.
- is a full-information game. Which means that both players always know everything about the position an played moves. No information is hidden to one or both players.
- is an impartial game. This means that in a certain position no matter which player is to move the set of possible (legal) moves is exactly the same. This for instance does not apply to chess, where players are only allowed to move their own colour pieces.
- is deterministic. No randomization is involved in the workings of the game.
- is finite. The game is guaranteed to finish within a finite number of moves and at each player's turn the number of moves to choose from is finite.

Feedback Vertex Kayles adheres to all these specifications, so Sprague-Grundy theory can be applied to it.

We will continue by giving the result of Sprague-Grundy theory important for this thesis. When the term game is use it refers to a game that adheres to the aforementioned specifications.

First let us define *mex* the *mex-rule*. For a finite set of nimbers S, the minimal excluded nimber of S, $mex(S)$, is defined as $mex(S) = min\{x \in \mathrm{N} | x \notin S\}$ with $mex(\emptyset) = 0$.

The mex-rule for general games states that the nimber for a position p is $mex(S)$ where S is the set containing all nimbers of positions reachable in one legal move from position p. Note that if there are no more legal moves from a certain position, S will be the empty set thus $nb(p) = 0$.

For Feedback Vertex Kayles we have that a move consists of removing one node from the graph. As will be further explained in Section 3.1 nodes that are not on a cycle can be removed from a graph. With all these nodes removed any node in the graph can be removed as a legal move. With this we can give a more formal definition of the mex-rule for Feedback Vertex Kayles. For this definition we define for graph $G = (V, E)$ and $v \in V$, $G - v = (V \backslash v, \{v'w | v' \neq v\})$

Definition 1. *For Feedback Vertex Kayles the* mex-rule *gives that given a graph $G = (V, E)$ with every node $v \in V$ on at least one cycle. If G is the empty graph $nb(G) = 0$ and otherwise $nb(G) = mex\{nb((V \backslash v, \{v'w | v' \neq v\}))\}$.*

Now that we have a rule to determine the nimber for a position, we get to what it means if a position has a certain nimber.

Theorem 1. *[Berlekamp et al., 1982] [Conway, 1976]. From a certain position there is a wining strategy for the first player to move if and only if the nimber for this position is greater than* 0*.*

Now let us define the *nim-sum*, denoted by $\oplus$; this is the binary sum without carry. It can be explained as the bitwise XOR on the nimbers represented as binary numbers. A certain position in the binary representation of the sum is 0 if both binary representations of the two nimbers have the same value for that postion, both 0 or both 1. Otherwise the position in the sum has value 1. More formal we have the following definition.

Definition 2. *The* nim-sum *of two nimbers* i_1 *and* i_2 *is* $i_1 \oplus i_2 = \sum\{2^j | i_1/2^j \textit{is even} \iff i_2/2^j \textit{is odd}\}$.

For example $9 \oplus 5 = 12$. When we use the binary representation this is easier to see: $1001 \oplus 0101 = 1100$.

With this definition we get to another interesting result of the theory: how games can be combined. If we have two game $\mathcal{G}_1$ and $\mathcal{G}_2$, we can get nimbers for the combination of these two games. The combination of two games $\mathcal{G}_1$ and $\mathcal{G}_2$ is a game in which on a turn a player can choose to make a move in either one of the two games. When in both games no more moves can be made the combined game comes to an end and the last player to have moved wins. We denote the combination of the two games $\mathcal{G}_1 + \mathcal{G}_2$. We denote a position in $\mathcal{G}_i$ by p_i and a position in the combined game $\mathcal{G}_1 + \mathcal{G}_2$ by (p_1, p_2). We denote the nimber of a specific position p by $nb(p)$.

Theorem 2. *[Berlekamp et al., 1982] [Conway, 1976]. Let* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *be two player full-information impartial deterministic finite games with a "last player to move wins"-rule. Let* p_1 *be a position in* $\mathcal{G}1$ *and* p_2 *be a position in* $\mathcal{G}_2$*. Now* $nb((p_1, p_2)) = nb(p_1) \oplus nb(p_2)$*.*

This theorem holds for any two games $\mathcal{G}_1$ and $\mathcal{G}_2$, so also for two instances of the same game. We can use this specifically for Feedback Vertex Kayles. When in Feedback Vertex Kayles a graph consists of two (or more) disjoint connected components, we can see these components as separate instances of the game. This can be done because the disjoint connected components have absolutely no effect on each other in the game. With this we can reformulate Theorem 2 to a form that is useful when analyzing Feedback Vertex Kayles. We get the following lemma.

Lemma 1. *Let* $G = G_1 \cup G_2$ *be a graph where* G_1 *and* G_2 *are two disjoint connected components of the graph. Now* $nb(G) = nb(G_1) \oplus nb(G_2$*.*

Because of Lemma 1 we know we can evaluate disjoint part of a graph separately and combine them using the nim-sum. For the general algorithm based on the mex-rule combined with Theorem 1 and described further in section 8 this can be really useful.

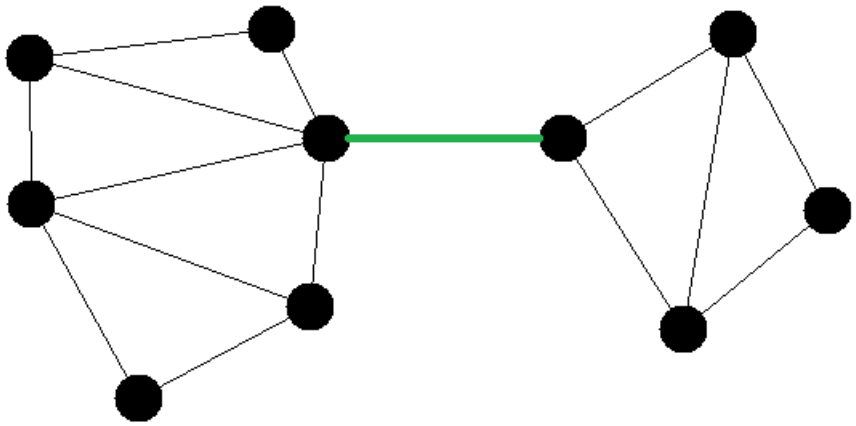

Figure 1: Example of a bridge (green)

2.3 Bridge

A *bridge* in a graph is an edge such that if the edge is removed the number of connected components in the graph increases by one. In other words a bridge is the only connection between two parts of the graph that would otherwise be separate connected components. An example can be seen in Figure 1. The green vertex in this graph is a bridge. When the edge is removed the nodes on the left and the nodes on the right form two separate connected components.

2.4 Chain

A chain is an interesting graph structure for Feedback Vertex Kayles. It is a chain of simple cycles connected through one common node for each pair of cycles. To make this definition more formal we have the following constructive definition.

Definition 3. *A graph is a chain if and only if it can be formed by the following rules:*

1. *Every graph consisting of exactly one simple cycle with all nodes on it is a chain.*

2. *If* $G_1 = \{V_1, E_1\}$ *and* $G_2 = \{V_2, E_2\}$ *are disjoint chains.* $\forall x_1, x_2 :$
   $G_3 = \{(V_1 \backslash x_1) \cup (V_2 \backslash x_2) \cup \{x_3\},$
   $\{e_{xy} \in E_1 | x \neq x_1\} \cup \{e_{xy} \in E_2 | x \neq x_2\} \cup \{e_{x_3y} | x_1y \in E_1\} \cup \{e_{x_3y} | x_2y \in E_2\}\}$
   is a chain if and only if no cycle in G_3 *has more than two nodes on more than one cycle and no node in* g_3 *has a degree greater than* 4*.*

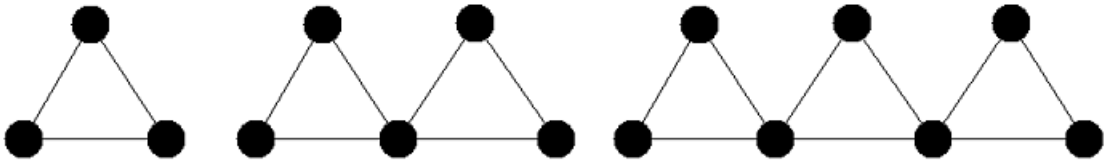

Figure 2: Examples of chains. From left to right: C_1, C_2, and C_3.

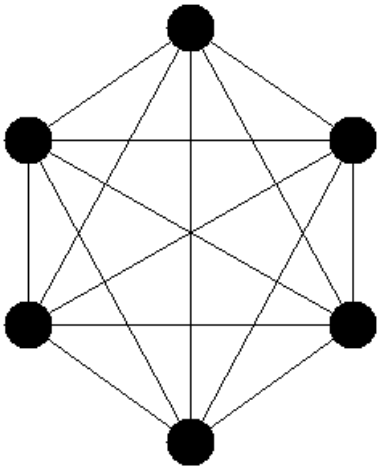

Figure 3: Examples of a complete graph with $n = 6$.

The length of a chain is defined as the number of cycles in a chain. We denote a chain of length x as C_x.

Figure 2 gives three examples of chains. These examples are the most simple examples of chains possible with each cycle containing exactly three nodes. As will be discussed in Section 3 these cycles are equivalent to cycles with more nodes only on one cycle. The most important lemma to support this claim is Lemma 2.

2.5 Complete graph

Definition 4. *A complete graph is a graph in which each vertex is connected to each other vertex by one edge.*

Figure 3 gives an example of a complete graph with 6 nodes ($n = 6$).

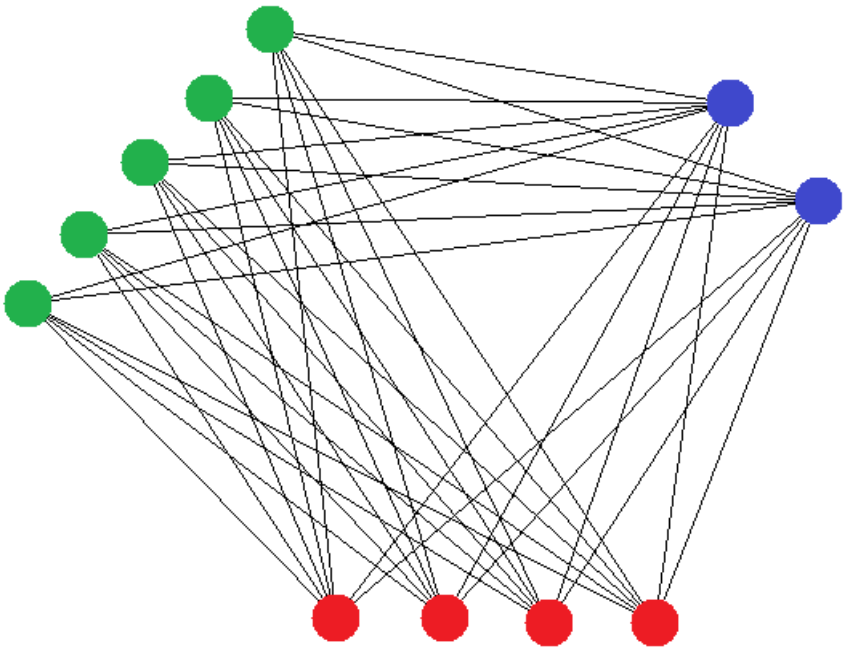

Figure 4: Example of a complete tripartite graph: $K_{2,4,5}$

2.6 Complete *k*-partite graph

A *complete k-partite graph* is a graph consisting of k independent sets where every node is connected to every other node that is not within the same independent set. The simplest and most commonly known k-partite graph is the bipartite graph consisting of two independent sets. The graph structure is interesting for Feedback Vertex Kayles and therefore we want an easy way to denote a k-partite graph.

A complete k-partite graph will in the sequal be denoted as $K_{a_1,a_2,...,a_k}$.Here a_1 to a_k are integers representing the number of nodes in each of the k independent sets. For example $K_{2,2,3,4,7}$ denotes a 5-partite graph with its sets containing 7, 4, 3, 2, and 2 nodes and $K_{5,8}$ denotes a bipartite graph with its sets containing 5 and 8 nodes. Figure 4 shows a visual example of a tripartite graph, specifically $K_{2,4,5}$. The three independent sets are marked by different colour nodes. It can be seen that each node in connected to every node from the two other sets but not to any node from its own set.

Note that there is no specific ordering in the individual independent sets, so we could write the integers for the sets in each order we want still denoting the same graph. For simplicity we will use the following notation rule: $a_1 \leq a_2 \leq ... \leq a_k$; this gives clarity and makes the notation easier to work with.

3 Basic insights

In this section some (almost) trivial but nevertheless important insights for Feedback Vertex Kayles are given and explained. These insights are useful both for a good understanding of the rest of the thesis as to improve (simple) algorithms for the problem.

3.1 Non-cycle nodes

Almost to trivial to mention is the fact that nodes that are on no cycle in the graph can be removed at any time without effecting the game. The nodes cannot be played, so they add nothing. When implementing any algorithm to find a winning strategy this should not be forgotten however. If an algorithm would consider non-cycle nodes as playable nodes this can influence the computed nimber for a specific graph.

3.2 Bridges

An important notion is that any bridge in a graph can be removed. As a bridge is an edge that is the only connecting edge between two otherwise separated connected components, it can never be on a cycle. Therefore when a bridge is removed this will never change the number of cycles in the graph, and thus it will not influence the game. As removing a bridge results in a graph with an extra connected component combined with Sprague Grundy theory removing bridges can potentially reduce computation time and space severely. The two resulting connected components can be evaluated separately and the nim-sum of the resulting two nimbers is the nimber for the entire graph.

For the removal of bridges to really be an addition to the algorithm, bridges have to be found with a fast algorithm. Assuming Feedback Vertex Kayles is PSPACE-complete, a polynomial time algorithm is sufficient. For removing bridges there is a fast polynomial algorithm found by [Tarjan, 1974]. For each connected component in the graph this algorithm first constructs a spanning tree for the component. Then it converts the spanning tree into a directed rooted tree. After this it computes some needed values for each node. Then by comparing these values for directly connected nodes, it can be determined which edges are bridges. All steps for the algorithm take $O(n + m)$ time, where n is the number of nodes and m is the number of edges. As for in every simple graph $m = O(n^2)$, the full algorithm takes $O(n^2)$ time.

3.3 Simple paths of degree-*2* nodes

A simple path on its own by definition of course only contains nodes with degree 2. With a simple path of degree-2 nodes we denote a path in a graph that contains only nodes that in the full graph have degree 2. Figure 5 shows an example of a graph with a path of degree-2 nodes. The green nodes in this figure are on the path of degree-2 nodes in the graph.

Lemma 2. *The set of nodes on a simple path of degree-*2 *nodes can be reduced to one node connected to the two nodes that were connected to the nodes on either endpoint of the path.*

Proof. If none of the nodes on the path is on a cycle all nodes can be removed from the graph without changing the nimber, so the lemma obviously holds. As all nodes on the path have degree 2 this path is the only simple path through the nodes. Because of this every simple cycle that goes through one of the nodes must also go through all of the other nodes on the path. In other words all nodes on the path are on exactly the same set of cycles. This means that regardless of which of the nodes on the path is removed, exactly the same cycles are broken. This also means that when one of the nodes on the path is removed the other nodes are no longer on any cycle. With all this we can conclude that reducing the set of nodes on the path to one node connected to the same nodes as the two outermost nodes on the path does not change the nimber of the graph and can thus always be done. □

Figure 6 shows the reduced version of the graph in Figure 5 in accordance with Lemma 2. The set of green nodes in Figure 5 is reduced to one green node in Figure 6. Both graphs are equivalent and have nimber 4.

When the representation of a graph contains for each node its degree, paths of nodes with degree 2 can be found rather easily. An algorithm can simply go over all nodes until a node with degree 2 is found. When such a node is found in both directions the next neighbour is marked found and if such a neighbour also has degree 2 the next neigbour is checked recursively. The recursion continues until finally a node with another degree is found and the recursion folds back. During the recursion the List of nodes on the path is recorded. After the recursion the loop over all nodes continues, but nodes that are already marked as found in an earlier recursion are not evaluated again. When the loop over all nodes is complete every path of nodes can be replaced by one node connected to the nodes that the outermost two nodes of the original path were connected to. As there cannot be more nodes on paths than there are nodes in total, this algorithm simply take $O(n)$ time with n being the number of nodes in the graph. There is one exception to the algorithm that should be handled. If and only if, under the assumption that the

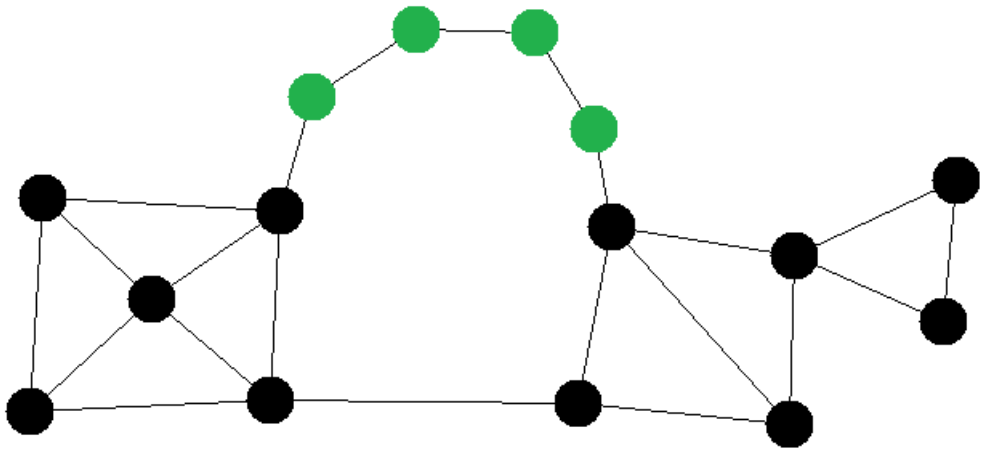

Figure 5: Example of graph with path of degree-2 nodes. The green nodes are on the path.

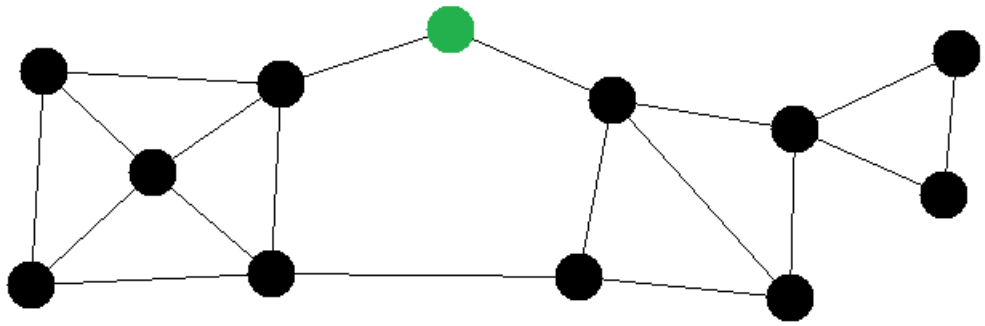

Figure 6: Reduced version of graph in Figure 5. The green node replaces the set of green nodes in the original graph.

graph is connected, all nodes are on one string the recursion will, after all other nodes, find the starting node for the recursion. In this case the string cannot be reduced to one node, but instead we know that the graph is only one simple cycle and thus the nimber for the graph is 1. This also means that no further evaluation is needed. Obviously in this case the algorithm still takes $O(n)$ time.

3.4 Isomorphic graphs

Isomorphic graphs are two distinct graphs for which each node in one graph can be mapped to a node in the other graph and vice versa in such a way that all mapped nodes are connected to exactly all the nodes on which the nodes are mapped that they were connected to in the other graph. Two isomorphic graphs have exactly the same nimber. When isomorphic graphs can be recognized in combination with memoisation this can reduce the number of possible positions to evaluate. It should be noted however that for general graphs recognizing isomorphic graphs is a hard problem.

3.5 Isomorphic nodes

Apart from full isomorphic graphs we define *isomorphic nodes*. We define these as two or more nodes which when removed from the graph result in graphs that are isomorphic to each other. The resulting graphs can be exactly the same graph, but also other isomorphisms. For these nodes we have that only one of the nodes has to be evaluated as removing the other nodes will simply move to a graph with the same nimber. In small graphs such nodes can often be spotted relatively easy by a human observer, especially when the graph is symmetric. In practice however for an algorithm these nodes can only be recognized by recognizing that the resulting graphs are isomorphic which just reduces the problem to recognizing isomorphic graphs.

3.6 Upper bound on nimbers

Although nimbers are simply integers in N and thus can be infinitely large, there are bounds on the nimber for a given graph. The first thing to note is that a nimber can never be larger than n where n is the number of nodes in the graph. As defined in Section 2.2 for a graph G we have $nb(G) = mex\{nb((V \backslash v, \{v'w | v' \neq v\}))\}$. As the set of nimbers defined in this definition contains the nimbers for all graphs with one unique node removed from G, there cannot be more distinct nimbers in the set than there are distinct nodes in G. For a nimber to be n we need that $mex(S) = n$, so S must contain all nimbers 0 through $n - 1$ which are exactly n distinct nimbers. This also means that a higher nimber greater than n would

require more than n distinct nimber in S which is impossible for a graph with n nodes. We can conclude from this that for any graph the nimber cannot be larger than n.

We can easily further lower this bound by 2. As we need at least three distinct nodes to form a cycle, graphs with $n = 1$ and $n = 2$ can never have a cycle and thus always have nimber 0. Only starting at $n = 3$ we can have a graph (the simple cycle containing all three nodes) with nimber 1. We also cannot have higher nimbers than 1. This is because from the simple cycle with three nodes only moves to graphs with two nodes and thus nimber 0 are possible. Of course there are also graphs with three nodes and no cycles which again have nimber 0. From graphs with four nodes only moves to graphs with 3 nodes can be made. As the nimber for graphs with three nodes are smaller or equal to 1, nimbers for graphs with $n = 4$ are smaller or equal to 2. We can continue this argumentation for every higher value of n. From this we can conclude that the nimber for a graph with n nodes is always smaller or equal to $n - 2$.

Until now we have only discussed general graphs and kept the possibility of all the moves leading to graphs with different nimbers. Often this shall not be the case and only few graphs will actually have a nimber equal to $n - 2$. For many graphs we can however directly state that the nimber will be significantly smaller than $n - 2$. Here we get back to isomorphic nodes as discussed in Section 3.5. As each group of isomorphic nodes result in an isomorphic graph (or even exactly the same graph) when moved upon, each group only adds one distinct nimber to the set of nimbers. Therefore the possible number of distinct nimbers of graphs reachable in one move is decreased by the number of isomorphic nodes minus the number of groups of isomorphic nodes. Let us denote the number of isomorphic nodes is a graph as nin and the number of groups of isomorphic nodes as gin. We can set the upper bound on nimbers for all graphs to $n-(nin-gin)$. Note that all three nodes in the graph with only a simple cycle of three nodes are isomorphic. Therefore the -2 in the last bound is contained in $-(nin-gin)$ in the new bound.

As also mentioned in Section 3.5 in general it is not easy to find isomorphic nodes as this requires to check that the graphs they lead to are isomorphic. However for some graphs we know certain nodes are ismorphic. The easiest example are complete graphs; here all nodes are ismorphic. As will be proven in Section 4.4 nimbers for fully connected graphs are smaller or equal to 1. This also follows from our bound $n - (nin - gin)$ which filled in for complete graphs becomes $n - (n - 1) = 1$. A second important case is further discussed in this thesis are complete k-partite graphs. As will also be discussed in section 4.5 all nodes in one independent set are isomophic. The upper bound on nimbers for k-partite graphs is thus $n - (n - k) = k$. Finally we have twins, which are discussed further in Section 6. All twins are isomorphic nodes and thus the upper bound of nimbers

for graphs with twins is $n - (ntwin - gtwin)$. Here $ntwin$ is the number of twins in the graph and $gtwin$ is the number of distinct groups of twins in the graph.

4 Nimbers for specific graph structures

4.1 Introduction to using nimbers

Just like with regular Kayles, nimbers can be assigned to a position in Feedback Vertex Kayles. In Kayles and Nimbers [Bodlaender and Kratsch, 2002] a nice list of six characteristics a game needs to have to be analyzed with Sprague-Grundy Theory. Which means that each to each position in the game a nimber can be assigned corresponding to a position in the game of Nim. Feedback Vertex Kayles has all six characteristics and thus we can also use Sprague-Grundy theory on it. In the same way as with Kayles we can use the nimbers to create an algorithm to determine whether a certain position has a winning strategy for the first player. It can be used to look at algorithms for the general case and for algorithms for certain graph classes.

It is interesting to look at some different simple graph structures and the nimbers they have. This could lead to insights into more complex graph structures and possibly faster algorithms for those structures.

In the rest of this section we will give certain specific graph structures and the nimbers we can assign to them. Note that in all these examples we do not consider nodes in a graph that are not on any cycle, as such nodes are not interesting for the game and can thus be removed without changing the nimber.

4.2 Nimbers for chains of cycles

Chains are defined in Section 2 in Definition 3. In this subsection we examine the nimbers of chains in Feedback Vertex Kayles.

When examining the game on chains it quickly becomes clear that in a chain there is always a winning strategy for the first player and thus any chain will have a positive nimber. We will give a more formal prove for this claim. Recall that the length of a chain is defined as the number of cycles in the chain. Also recall that we denote a chain of length x by C_x.

Lemma 3. *Any chain of cycles C_x has a winning strategy for the first player.*

Proof. For C_1 any node can be removed to remove the only cycle in the graph and thus win the game. For C_2 any the node connecting the two cycles can be played. This also leaves no cycles in the graph. For C_x with $x \geq 3$ and $x \mod 2 = 1$ a node on the middle cycle and no other cycle can be played. This splits the chain

into two equal length chains. From here on the first player can simply mirror any move played on the second player on one of these chains, on the other chain. This can be done until the second player removes the last cycles from one chain in which case the first player can then remove the last cycles on the other chain. In this way the first player wins the game. For C_x with $x \geq 3$ and $x \mod 2 = 0$ the node connecting the two middle cycles can be played. This will split the chain into two chains of equal length. Again the first player can mirror the second players moves until the game is won. □

Although we know every chain has a positive nimber, this does not mean that every chain has the same nimber, or even that there is a simple regular pattern in the nimbers of chains of certain length. There is a regular pattern in the nimbers though. Although there is no literature on the Feedback Vertex Kayles game, the specific case with a chain of cycles is similar to another game on which there is more literature. As mentioned before Node Kayles was based on the old game of Kayles that is described in the introduction. When looking closely at Feedback Vertex Kayles on chains of cycles, we can see it behaves in exactly the same way as the original game of Kayles. Removing a node that is only on one cycle effectively removes that one cycle. If it is a cycle that is not on one of the ends of the chain, it splits the chain into two chains. This is equivalent to bowling one pin in Kayles. There also the row of pins is split into two rows if the bowled pin was not at one of the ends of the row. Removing a node that is on exactly two cycles effectively removes those two cycles and again possibly splits the chain into two chains. This is equivalent to bowling two adjacent pins, which can also split the row. Both with chains in Feedback Vertex Kayles and rows in Kayles it is impossible to remove two cycles or pins from separate chains or rows. The games thus behave exactly the same. Both Feedback Vertex Kayles on chains of cycles and original Kayles therefore have the same sequence of nimbers corresponding to lengths of chains and rows respectively. [Guy and Smith, 1956] have given the periodicity of these nimbers along with other results on mathematical games. They also use a more general proof they have constructed to prove the periodicity for Kayles.

4.3 Nimbers for sunflower graphs

There are different definitions of sunflower graphs in literature. Let us first give the definition of sunflower graphs as used in this thesis.

Definition 5. *A graph is a* sunflower graph *of size k, with $k \geq 3$ if it has $2k$ nodes and has the following structure. It has one cycle of k nodes which we call the inner ring. We number the nodes in the inner ring $1, 2, ..., k$, then for $1 \leq i \leq k-1$ we have a node $k+i$ connected with an edge to node i and $i+1$. Even though nodes*

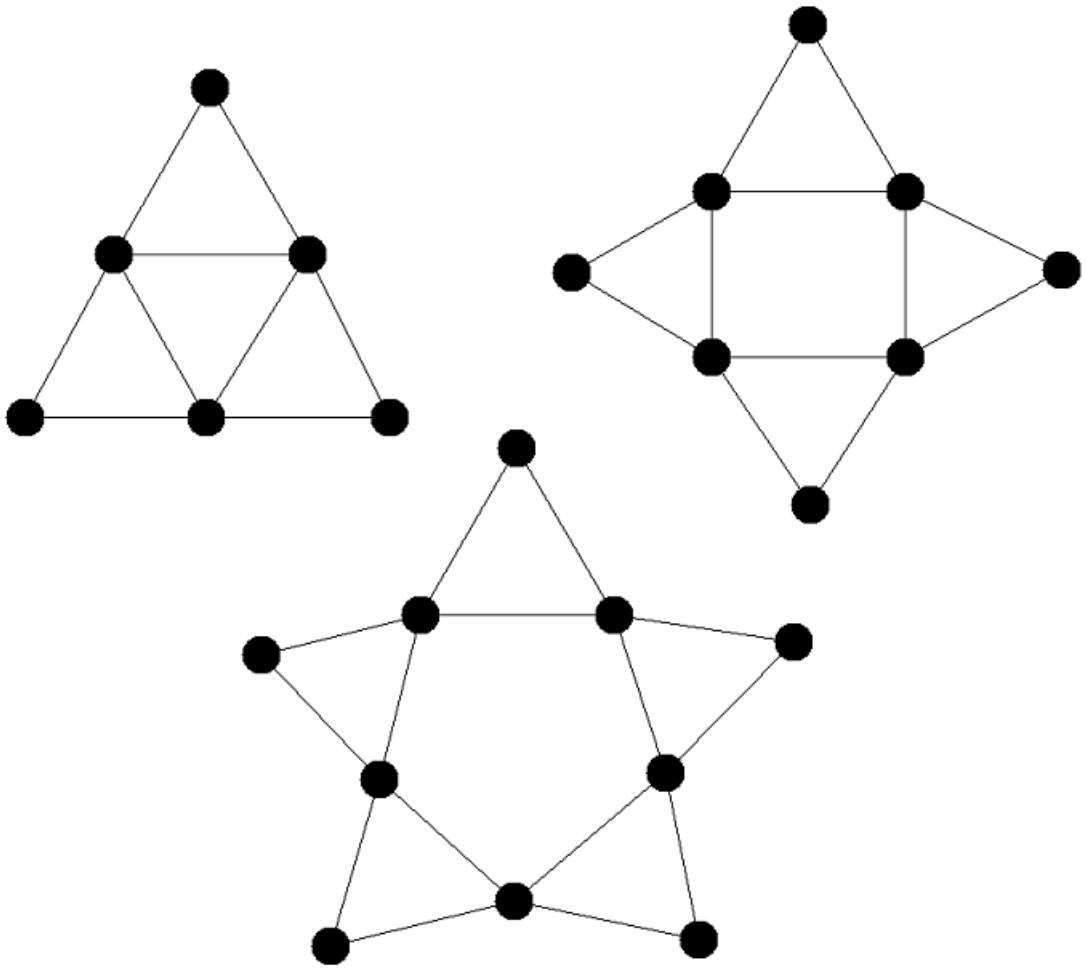

Figure 7: Examples of sunflower graphs. top left: S_3, top right: S_4, bottom: S_5

$k+1$ to $2k$ are not directly connected to one another, we call the set of these nodes the outer ring. There are no other edges in the graph.

We denote a sunflower graph by S_k.

In Figure 7 three examples of sunflowers are given S_3, S_4 and S_5.

Note that although not directly clear from the given definition, a sunflower graph is equal to a chain, as described in the previous section, of which the first and the last cycle are also connected by merging one of their independent nodes. Just like with chains we can note that the nodes that are only on one cycle are equivalent to a whole sequence of connected nodes that fill up a larger cycle, as all these nodes would be equivalent. Again for the sake of simplicity we only discus the minimal case.

Although the graph structure seems an interesting case for Feedback Vertex

Kayles at first glance, when studying it better we find that a graph with this structure always has the nimber 0 and thus always is a losing position for the first player. We can proof this looking at the two cases of a sunflower graph with a middle wheel with either an even or an odd number of nodes. Note that a graph has a nimber 0 if and only if there is no winning strategy for the first player, which in turn means that for any move the first player makes in the resulting graph there is a winning strategy for the second player.

Before we give the proof let us define a lane in a sunflower graph, as this will make the proof easier to read. In practice a lane is an uninterrupted chain in a graph.

Definition 6. *A lane is a set of* $2k-1$ *nodes in a graph such that for* $1 \leq i \leq k$*, node* i *is connected by an edge to node* $i+1$ *and for* $k+1 \leq j \leq 2k-1$ *node* j *is connected to node* $j-k$ *and node* $j-k+1$ *and there are no other edges between nodes in the set. By this definition a lane with* $2k-1$ *nodes has* $k-1$ *cycles in its induced subgraph. To keep close to the definitions for the sunflower graph we call nodes* 1 *to* k *inner ring nodes and nodes* $k+1$ *to* $2k-1$ *outer ring nodes. We call nodes* 1 *and* k *side inner nodes and we call nodes* $k+1$ *and* $2k-1$ *side outer nodes. These nodes are only on the cycles on the far sides of our lanes. We call all other nodes body nodes on top of their inner and outer nomination.*

Note that if we remove one node from the outer ring of a sunflower graph of size k, it contains one big lane of size $2k-1$. Furthermore note that in a sunflower graph with one or more outer ring nodes removed, the only thing keeping lanes from being chains, is the inner ring that forms an extra cycle.

Theorem 3. *Any sunflower graph has nimber 0.*

Proof. For this prove recall that the size of a sunflower graph is half the number of its nodes.

We will prove this theorem by proving two separate cases separately. First we will prove any sunflower graph of odd size has nimber 0. Then we will prove every sunflower graph of even size has nimber 0. Each is proven by showing that any move made by the first player has a response that ultimately leads to the second player winning.

For a sunflower graph of odd size there are two cases to distinguish for the first move of the first player. He either chooses a node from the inner ring or a node from the outer ring. Without loss of generality we can say for a sunflower graph of size k, he either chooses node 1 or node $k+1$. If the player chooses node 1 the resulting structure is a chain (larger than size 0). As we have shown a chain always has a positive nimber, and thus the resulting position has a winning strategy for the second player. If he chooses node $k+1$, the rest of the nodes form

a lane of size $2k-1$, which has $k-1$ cycles. As k is odd, we know that $k-1$ is even. Furthermore we know by the definition of a lane, that any body inner ring node is on exactly two cycles belonging to the lane, thus by removing such a node two cycles are removed from that lane. Besides that removing any inner ring node breaks the inner ring cycle and thus makes all maximal size lanes in the graph become separate chains. As the lane in the graph resulting after the first move has an even number of cycles, we know that there are two cycles that have evenly many cycles to either side of the pair of them. Removing the body inner ring node connecting this pair of cycles thus results in a graph with two equally long (0 long if k was 3) chains, which is a losing position. We can conclude that no matter what move the first player makes on his first turn, the second player has a winning counter. Thus for any sunflower graph of odd size the nimber is 0.

For a sunflower graph of even size we define the same two cases for the first player's first move: for size k he either chooses node 1 or node $k+1$. Again choosing node 1 on the first move result in a chain, which is a winning position for the second player. This time however the choice for node $k+1$ gives a harder position to solve; as the resulting lane has an odd number of cycles now, we cannot remove an inner node to create two chains of equal length. Instead the second player plays the node opposite to the node the first player played. More precisely he plays node $k+1+k/2$. The resulting graph contains exactly two lanes of maximal length, which are equally long. We pair these two lanes and call each the partner lane of the other. Now we have three cases for different moves of the first player. We will describe the cases, the response of the second player and the resulting graph.

1. The first player chooses a body outer ring node of any lane l. This splits lane l into two separate lanes s and p. The second player now chooses the corresponding note in the partner lane of l, which we call l', that splits l' into two lanes s' and p' such that s and s' are of equal length and p and p' are of equal length. Now l and l' are gone, but we pair up s and s' and p and p' as two new pairs of lanes. In the resulting graph we still only have pairs of lanes of equal length and the same three cases apply again.

2. The first player chooses a side outer ring node of any lane l. This removes one cycle from the side of lane l, any other pairs of lanes are unchanged. Here the second player chooses an side inner ring node of the partner lane of l, l'. As it is a side inner ring node, it is only on one cycle in this lane. As lanes are separate it is also on no cycle in any other lane than l'. Thus by removing the node one cycle is removed from l'. Moreover because it is an inner node, the inner cycle is broken and thus all lanes become chains. As in every pair of lanes the lanes had equal length, these pairs become pairs of chains of equal length. As from both l and l' one cycle is removed, the two resulting chains are also of equal length (possibly length 0). With all pairs

of chains of equal length the resulting graph, in which the first player is to move, is a losing position.

3. The first player chooses any inner ring node of any lane l. The move breaks the inner cycle and thus turns all lanes into chains. All pairs of lanes except for l and its partner lane l' become pairs of chains of equal size. Furthermore the move either creates one chain c with one less cycle than lane l, or splits lane l and creates two separate chains s and p. The second player plays the node on l', which is now a chain, that corresponds to the node player 1 played on l such that either this creates a chain c' with equal length to c or this creates two chains s' and p' with the length of s equal to the length of s' and the length of p equal to the length of p'. Thus creating a position with only pairs of chains of equal length, which is a losing position.

Note that these three cases describe all possibilities. The keen observer will see that there is no case where the first player chooses a node that is not on any lane. This however is because as long a the first player only plays according to Case 1 there are no nodes that are not on any lane. As any body inner ring node is connected to two outer ring nodes, one move on a body outer ring node cannot make it not lay on a lane anymore. A side inner ring node is only connected to one outer ring node, so removing that node makes it no longer lay on any lane, but that outer ring node must by definition be a side outer ring node, which when played falls under Case 2. Finally the only way for an outer ring node to no longer be on any lane is for any inner ring node to be removed. Playing such a node falls under Case 3. Both Case 2 and 3 lead to a position in which the second player directly plays a move to get into a clearly lost position for the first player and thus does not continue to cases for the first player. So these three cases cover all possibilities.

Before concluding the proof that sunflower graphs of even size always have nimber 0, we need to prove that playing according to the three given cases always leads to Case 2 or 3 within a finite number of moves. This is easily done if we look at the characteristics of lanes and the behavior of Case 1. For any lane with with a body outer ring node, there must be two side cycles on the lane and at least one body cycle. So any lane with only one or two cycles has only side outer ring nodes. Playing according to Case 1 makes lanes l and l' split into two lanes as one of the cycles is removed by playing on the lane and each of the lanes resulting from the split has at least length 1 (otherwise no body node but a side node was selected). The maximum length of any of the resulting lanes is 2 smaller than the length of l. Since the original lanes are of finite length, there is only a finite amount of times lanes can be split in this way, until all lanes in the graph have a length of maximally 2. At this point any outer ring node is a side outer ring node and thus the only possible moves for player 1 fall under Case 2 or 3.

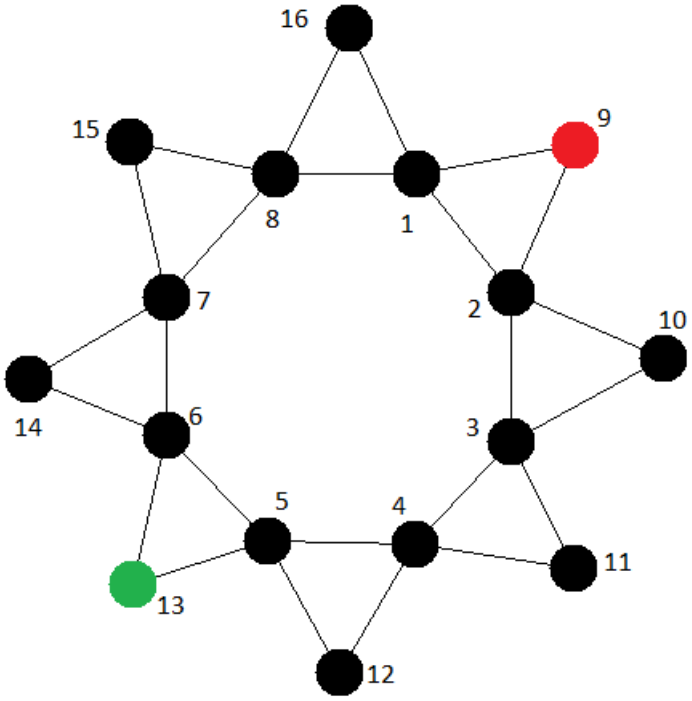

Figure 8: Example illustrating the proof for Theorem 3; graph S_8. The nodes are numbered according to the definition of sunflower graphs. According to the second option for the first move, player 1 moves on the red node (9). Player 2 responds according to the strategy by playing the green node (13).

This leads to the conclusion that any sunflower graph of even size has nimber 0. Combining our results we can conclude that any sunflower graph has nimber 0. □

As the part of the proof for sunflower graphs of even size is rather complex, in Figures 8 through 12 graphs are depicted for the most important steps in the strategy. As the reaction to a move on the inner lane is most intuitive these cases are not depicted. In Figure 8 the starting position for S_8 is depicted. In Figure 9 we see the situation after two moves (one for each player) and the the moves and reactions for Case 1 and 2. In Figure Figure 10 we see the situation after a move according to Case 1 in Figure 9, with the move and reaction for Case 2 depicted. In this situation a move according to Case 1 is no longer possible. In Figure 11 we see the situation after a move according to Case 2 in Figure 9, which is clearly a losing position for player 1. In Figure 12 we see the situation after the moves from Figure 10, which is also clearly a losing position for player 1.

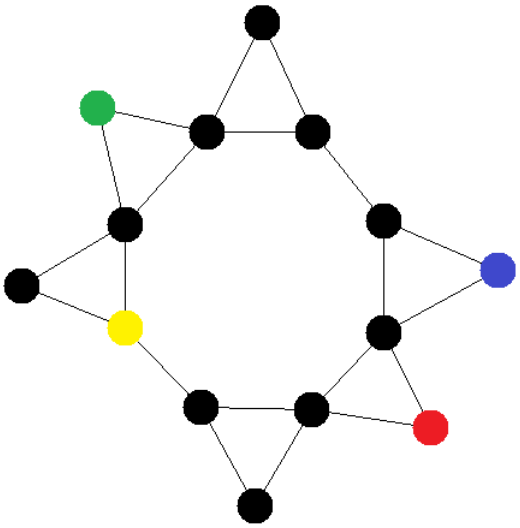

Figure 9: Situation after first two moves. The red node depicts the first player's move for Case 1 and the green node depicts the reaction. The blue node depicts the first player's move for Case 2 and the yellow node depicts the reaction.

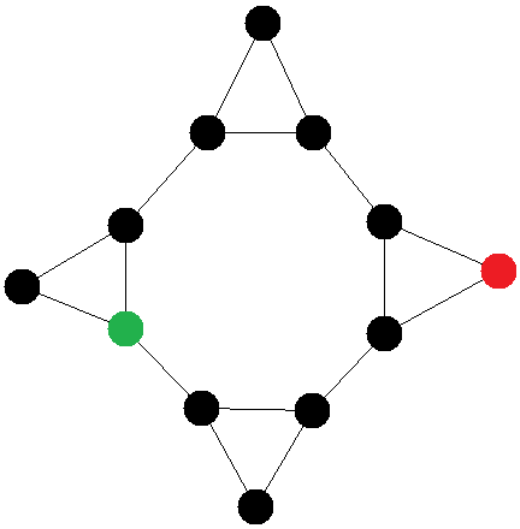

Figure 10: Situation after four moves in Case 1. The red node depicts the first player's move and the green node depicts the reaction.

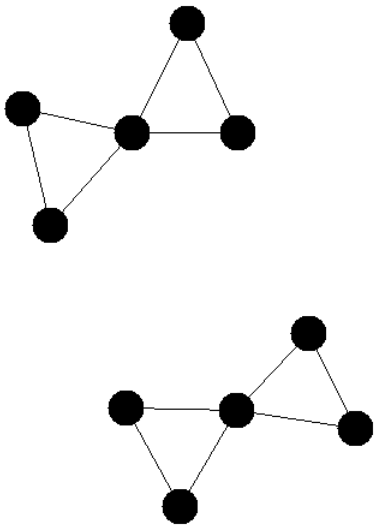

Figure 11: Situation after four moves in Case 2. This graph consisting of two instances of C_2 is clearly a losing position for the first player.

Figure 12: Situation after six moves, first Case 1, then Case 2. This graph consisting of two instances of C_1 is clearly a losing position for the first player.

4.4 Nimbers for complete graphs

Complete graphs are defined in Section 2 in Definition 4. In a complete graph larger than two nodes every node is on a cycle; as all nodes are connected to all nodes there is always a simple path from one node through at least two other nodes back the the first node. When removing one node from a complete graph, the resulting graph also is a complete graph. Therefore on a complete graph all but two nodes have to be played before the game is over. The number of moves needed however does not relate to the difference in nimbers that can be assigned to complete graphs. These graphs increasing in size alternately have the nimbers 0 and 1.

Lemma 4. *Complete graphs of odd size have nimber* 1.

Lemma 5. *Complete graphs of even size have nimber* 0.

Proof. We will proof both lemmas in one inductive proof with induction to n.

Induction hypothesis: All complete graphs of odd size up to size $n-2$, with $n \mod 2 = 1$, have nimber 1 and all complete graphs of even size up to size $n-1$ have nimber 0.

Base cases: On a complete graph of three nodes playing either one node results in a graph with no more cycles (nimber 0), so the nimber for this graph is 1.

On a complete graph of four nodes playing either one node results in a complete graph with three nodes, which we have just shown to have nimber 1. So this graph has nimber 0.

Induction step: Playing any node on the complete graph of size n results in a complete graph of size $n-1$. As n is odd, $n-1$ is even and thus the resulting graph has nimber 0. So the complete graph of size n has nimber 1. Playing any node on the complete graph of size $n+1$ results in a complete graph of size n, which we have just shown to have nimber 1. So the complete graph of size $n+1$ has nimber 0.

□

4.5 Nimbers for complete k-partite graphs

In Section 2, complete k-partite graphs are defined. It is already mentioned there that they are interesting for Feedback Vertex Kayles and this subsection will discuss them. Although not as simple as with normal complete graphs, on fully

connencted k-partite graphs Feedback Vertex Kayles can be computed easier than on general graphs. This however is under the assumption that the general problem is indeed PSPACE-complete. The following subsections will discuss different parts of the problem with k-partite graphs. In Section 4.5.1, some basic but important insights about the important specifications of k-partite graphs will be given. These insights are an important basis for the rest of the methods and proofs used in the rest of the subsections. This will be followed by Section 4.5.2 on k-partite graphs in general. For which a general algorithm is given and analyzed. Then in Section 4.5.3 and 4.5.4 the special cases of bipartite and tripartite graphs are discussed and even faster solutions are given for these cases. In Section 4.5.5 it will be discussed how the solutions for bipartite and tripartite graphs can be used to improve the general algorithm. Finally in Section 4.5.6 some closing remarks will be given on complete k-partite graphs.

4.5.1 Basic insights

Before we start to go deeper into the different values for k and algorithms to compute nimbers for the resulting graphs, it is important to understand the specific properties of these graphs concerning Feedback Vertex Kayles. As is also discussed further in Section 6, nodes that are connected to exactly the same nodes are equivalent. So only one of these nodes has to be evaluated each step. If we take a look at k-partite graphs we can see that all nodes within one independent set are twins and thus equivalent. Furthermore there is no important distinction between the different independent sets; a graph with four nodes in the first set, two nodes in the second set, and three nodes in the third is equivalent to one with three nodes in the first set, four nodes in the second set and two nodes in the third. Knowing this we can represent the graph as a set of k integers. The possible moves on a graph then correspond with lowering one of these k integers by one. When an integer is reduced to zero by this means, the last node of the corresponding independent set is removed and the resulting graph is a $(k-1)$-partite graph.

As long as there are at least three sets containing nodes remaining in the graph there is always a cycle in the graph. The smallest case possible with three sets, $K_{1,1,1}$, is a simple cycle with all three nodes on it. As adding more nodes to any graph and especially these graphs does never remove any cycle, it is obvious that any larger case also contains at least one cycle.

When there are only two sets containing nodes in a complete k-partite graph, which is thus a bipartite graph, when both sets have at least two nodes there is always a cycle in the graph. The smallest case $K_{2,2}$ is one simple cycle containing all four nodes. Again as adding more nodes does not remove cycles any case with more nodes per set also contains at least one cycle.

Even though $K_{2,2}$ already has a cycle, when we have two sets and one of them

contains only one node, there will never be a cycle in the graph.

Lemma 6. *For all i the nimber of $K_{1,i}$ equals 0.*

Proof. Every node in the set with i nodes can be reached from any other node by a path through the single node in the other set, but there are no other connections in the graph to complete a cycle. As there are no cycles in the graph there is no legal move for the first player and thus it is a losing position and has nimber 0. □

4.5.2 Nimbers for complete k-partite graphs

The insights in the previous subsection give us all the handles we need to construct an algorithm solving any case of a complete k-partite graph with a faster algorithm than for general graphs. In this algorithm we still go through every possibility, but the notions from the last subsection really limit the number of possibilities.

Before starting the main algorithm the graph should be converted to a set of integers representing for every independent set in the graph the number of nodes in it. Note that this integer set really is a set as the order of the integers does not matter. We will use a hashtable with sets as keys and nimbers (integers) as values, to keep track of the already computed sets of integers. It is important that the objects representing the sets are either compared in such a way that sets with the same integers but in a different order are really recognized to be the same, or upon creation as an object are made to be exactly the same. Of course we can simply sort the integers in $O(k \log k)$ time, which is not significant for the bound on the complete algorithm.

In Algorithm 1 the algorithm to compute the nimber for a general complete k-partite graph is given in pseudocode. The main function is ComputeNimber. As its input it expects a k-partite graph. The not specified function makeSet constructs an integer set from a given k-partite graph corresponding to it. The function also determines the values of k and $maxSize$. Here k is the number of independent sets and $maxSize$ is the number of nodes in the largest set. A hashtable with integer sets as keys and integers as values is created. All combination of 1 and the integers 1 through $maxSize$ are added to the hashtable with nimber 0 corresponding with Lemma 6. These combination are the endpoints for the recursion.

With everything ready the RecurseNimber function is called which will recursively find the nimber for the graph. If the specific set the function gets as argument has already been evaluated the nimber will be in the hashtable and is returned immediately. If not the nimber is computed recursively. The mex-rule is used with all nimbers for situations reached within one move from the current graph (Set). For every integer in the set a possible move is to subtract 1 from the integer and keep all other integers the same. If an integer would become 0 in this way this would mean the corresponding independent set would be fully

Algorithm 1 Compute Nimber k-parite

```
1: function ComputeNimber(graph Start)
2:     int k, maxSize;
3:     Set startSet = makeSet(start, out k, out maxSize);
4:     Hashtable nimbers = new Hashtable<Set, int>;
5:     fillHash(k, maxSize, nimbers);
6:     return RecurseNimber(k, maxSize, startSet);
7: end function
8:
9: function RecurseNimber(int k, int maxSize, Set startSet)
10:    int result;
11:    if (result = nimbers.getValue(startSet)) == null then
12:        List<Int> options = new List;
13:        for all int part in startSet do
14:            if part == 1 then
15:                newSet = startSet.remove(part);
16:            else
17:                newSet = startSet.replace(part, part -1);
18:            end if
19:            options.Add(RecurseNimber(k, maxSize, newSet);
20:        end for
21:        result = MEX(options);
22:        nimbers.Add(startSet, result);
23:    end if
24:    return result
25: end function
26:
27: function fillHash(int k, int maxSize, Hashtable nimbers)
28:    for int i = 1; i ≤ maxSize; i++; do
29:        nimbers.Add(Set(1,i), 0);
30:    end for
31: end function
```

Table 1: Nimbers for complete bipartite graphs with $p + q$ nodes

p	q	nimber	rules
1	x	0	
2	$x = 2y$	1	$x \geq 2$
2	$x = 2y - 1$	2	$x > 2$
x	x	0	$x > 2$
3	$x = 2y$	2	$x > 3$
3	$x = 2y - 1$	0	$x > 3$
$x = 2y$	$z = 2w$	0	$x > 3, z > x$
$x = 2y$	$z = 2w - 1$	1	$x > 3, z > x$
$x = 2y - 1$	$z = 2w$	1	$x > 3, z > x$
$x = 2y - 1$	$z = 2w - 1$	0	$x > 3, z > x$

removed from the graph. In that case no more moves are possible on that specific independent set, so the integer is removed from the integer set instead of being decremented. In an implementation it would of course also be possible to work with zeros and not considering these integers for a move anymore. When the nimber for a specific integer Set is found it is added to the hashtable so it will not be evaluated again.

Using this algorithm every unique combination of integers where every integer is smaller or equal to the integers in the first set is evaluated exactly once. With this an easy but not very restricted upper bound is $O((n/k)^k)$. From this we already have that the algorithm is polynomial in the number of nodes n. Note that the algorithm is exponential in the number of independent sets k. However, when k is significantly large in comparison to n, n/k will be small and thus the complexity will still be limited.

4.5.3 Nimbers for complete bipartite graphs

Looking at the nimbers for complete bipartite graphs we can see that we do not need the algorithm from the last subsection to compute the nimber for a specific graph in this group. Instead there is a regularity of which we only have to implement the rules to be able to compute the nimber of any bipartite graph.

Table 1 displays all the rules to describe the regularity of nimbers for complete bipartite graphs. Note that x, y, z and w are all integers.

This table cannot only be derived from observation but can also be constructed through a series of inductive proofs. To maintain an accessible proof for the whole set of rules, we will separate the prove into separate lemmas with their own proof. These lemmas are finally combined to prove the entire theorem.

Lemma 7. *$K_{2,2}$ has nimber 1.*

Proof. As both sets contain exactly two nodes, all possible moves on the graph are equivalent. The resulting graph from a move is $K_{1,2}$. By Lemma 6 we know this resulting graph to have nimber 0. Using the mex-rule we conclude that the original graph has nimber 1. □

Lemma 8. *$K_{2,3}$ has nimber 2.*

Proof. The two possible distinct moves in this case are removing a node from the set with two nodes and removing a node from the set with three nodes. The first possible move results in $K_{1,3}$ which by Lemma 6 has nimber 0. The second possible move results in $K_{2,2}$ which by Lemma 7 has nimber 1.

Using the mex-rule we conclude that the original graph has nimber 2. □

Lemma 9. *$K_{2,q}$ has nimber 1 if q is even and $q \geq 2$, and has nimber 2 if q is odd and $q > 2$.*

Proof. We give an inductive proof for this lemma with induction to q.
Induction hypothesis:
For every $q' < q$ the nimber for the graph is 1 if q' is even and $q' \geq 2$, and is 2 if q' is odd and $q' > 2$.

Base Cases:
Our base cases are given by Lemmas 7 and 8, which correspond to the cases $q = 2$ and $q = 3$.

Induction step:
From $K_{2,q}$, we can move to $K_{1,q}$, or to $K_{2,q-1}$. Regardless of whether q is even or odd, the nimber for the first possibility is 0, following Lemma 6. For the second situation if q is even, $q - 1$ is odd, so the nimber for the resulting graph is 2, and if q is odd $q - 1$ is even, so the nimber for the resulting graph is 1. Using the mex-rule we get that if q is even the nimber is 1 and if q is odd the nimber is 2. □

Lemma 10. *Let $x \geq 3$. The nimber of $K_{x,x}$ is 0.*

Proof. If both sets contain exactly three nodes every possible move will result in the graph $K_{2,3}$, which following Lemma 8 has nimber 2. Using the mex-rule we get that the original graph has nimber 0. If $x > 3$ a simple strategy for the second player is to simply play a node on the other set than the set of which the first player has removed a node, resulting in a graph with two sets containing exactly $x-1$ nodes, until the graph with two sets containing three nodes is reached. As we have just shown that this last position has nimber 0 this means that every higher value for x also gives a graph with nimber 0. □

Lemma 11. *$K_{3,4}$ has nimber 2.*

Proof. The two possible distinct moves in this case are removing a node from the set with three nodes and removing a node from the set with four nodes. The first possible move results in $K_{2,4}$ which by Lemma 9 has nimber 1. The second possible move results in $K_{3,3}$ which by Lemma 10 has nimber 0.

Using the mex-rule we conclude that the original graph has nimber 2. □

Lemma 12. *$K_{3,q}$ has nimber 2 if q is even and $q > 3$, and has nimber 0 if q is odd and $q \geq 3$.*

Proof. We give an inductive proof for this lemma with induction to q.
Induction hypothesis:
For every $q' < q$ the nimber for the graph is 2 if q' is even and $q' > 3$, and is 0 if q' is odd and $q' \geq 3$.

Base Cases:
Our base cases are given by Lemmas 10 and 11, which correspond to the cases $q = 3$ and $q = 4$.

Induction step:
From $K_{3,q}$ we can move to $K_{2,q}$ or to $K_{3,q-1}$. In the first situation by Lemma 9 we have that the nimber of the resulting graph is 1 if q is even and 2 if q is odd. For the second situation if q is even, $q-1$ is odd, so the nimber for the resulting graph is 0, and if q is odd $q-1$ is even, so the nimber for the resulting graph is 2.

Using the mex-rule we get that if q is even the nimber is 2 and if q is odd the nimber is 0.

□

Lemma 13. *$K_{4,5}$ has nimber 1.*

Proof. The two possible distinct moves in this case are removing a node from the set with four nodes and removing a node from the set with five nodes. The first possible move results in $K_{3,5}$, which by Lemma 12 has nimber 0. The second possible move results in $K_{4,4}$, which by Lemma 10 has nimber 0.

Using the mex-rule we conclude that the original graph has nimber 1. □

Lemma 14. *$K_{4,q}$ has nimber 0 if q is even and $q \geq 4$, and has nimber 1 if q is odd and $q > 4$.*

Proof. We will give an inductive proof for this lemma with induction to q.
Induction hypothesis:
For every $q' < q$ the nimber for the graph is 0 if q' is even and $q' \geq 4$, and is 1 if

q' is odd and $q' > 4$.

Base Cases:
Our base cases are given by Lemmas 10 and 13, which correspond to the cases $q = 4$ and $q = 5$.

Induction step:
From $K_{4,q}$ we can move to $K_{3,q}$ or to $K_{4,q-1}$. In the first situation by Lemma 12 we have that the nimber of the resulting graph is 2 if q is even and 0 if q is odd. For the second situation if q is even, $q-1$ is odd, so the nimber for the resulting graph is 1, and if q is odd $q-1$ is even, so the nimber for the resulting graph is 0.

Using the mex-rule we get that if q is even the nimber is 0 and if q is odd the nimber is 1.

□

Lemma 15. *Let* $4 \leq p \leq q$. $K_{p,q}$ *has nimber* 0 *if* $p \mod 2 = q \mod 2$ *and nimber* 1 *otherwise.*

Proof. We give an inductive proof for this lemma with induction to $p + q$.
Induction hypothesis:
Let $p + q = x$ and $p' + q' = x'$. For every $8 \leq x' < x$, $K_{p',q'}$ has nimber 0 if $p' \mod 2 = q' \mod 2$ and nimber 1 otherwise.

Base Case:
As base case we have $K_{4,4}$, by Lemma 10 this graph has nimber 0 which corresponds to this lemma.

Induction step:
We have two cases, $p = 4$ and $p > 4$.

Case 1. $p = 4$**:** From Lemma 14 we have that if q is even and thus $p \mod 2 = q \mod 2$ the nimber for the graph is 0 and that is q is odd and thus $p \mod 2 \neq q \mod 2$ the nimber is 1.

Case 2. $p > 4$**:** As $p \leq q$ we have that $q > 4$. From $K_{p,q}$ we can go to $K_{p-1,q}$ or $K_{p,q-1}$, where both $p - 1 \geq 4$ and $q - 1 \geq 4$. If $p \mod 2 = q \mod 2$ then $p - 1 \mod 2 \neq q \mod 2$ and $p \mod 2 \neq q - 1 \mod 2$ thus both $K_{p-1,q}$ and $K_{p,q-1}$ have nimber 1. If $p \mod 2 \neq q \mod 2$ then $p - 1 \mod 2 = q \mod 2$ and $p \mod 2 = q - 1 \mod 2$ thus both $K_{p-1,q}$ and $K_{p,q-1}$ have nimber 0.

Using the mex-rule we get that if $p \mod 2 = q \mod 2$ the nimber is 0 and otherwise the nimber is 1. □

Theorem 4. *Given a graph $K_{p,q}$, the nimber for the graph is given by table 1.*

Proof. To start with for a graph to be truly denoted as a bipartite graph ($K_{p,q}$), the number of nodes in both sets should be larger than 0. Therefore cases with less than 1 node in a set are not in the table (although they obviously have nimber 0).

Without loss of generality we can assume that $q \geq p$ as the sets are equivalent and do not need a specific ordering, so we can choose to assign the values of the number of nodes in the sets to p and q in such a way that $q \geq p$ without changing the problem. With this we have that all combinations of numbers of nodes in both sets are incorporated in the table.

The table's first row follows from Lemma 6.

The table's second and third row follow from Lemma 9.

The table's fourth row follows from Lemma 10.

The table's fifth and sixth row follow from Lemma 12.

Finally the last four rows of the table follow from Lemma 15. □

4.5.4 Nimbers for complete tripartite graphs

Following from the proved rules for bipartite graphs, an interesting question that arises is whether a similar set of rules can be proven for tripartite graphs. This turns out to be the case, although it requires double the amount of rules to get all cases and also needs the rules for bipartite graphs as basis.

Lemma 16. *$K_{1,1,1}$ has nimber 1.*

Proof. Any possible move result in $K_{1,1}$ which by Lemma 6 has nimber 0. Using the mex-rule we get that $K_{1,1,1}$ has nimber 1. □

Lemma 17. *$K_{1,1,2}$ has nimber 2*

Proof. There are two possible moves, either to $K_{1,1}$ or to $K_{1,1,1}$. The first case by Lemma 6 has nimber 0. The second case by Lemma 16 has nimber 1.

Using the mex-rule we get that $K_{1,1,2}$ has nimber 2. □

Lemma 18. *Let $r \geq 1$, $K_{1,1,r}$ has nimber 1 if r is odd and 2 otherwise.*

Proof. We will give an inductive proof for this lemma with induction to r.
Induction hypothesis:
For every $r' < r$, $K_{1,1,r'}$ has nimber 1 if r' is odd and nimber 2 otherwise.

Base Cases:
The base cases are provided by Lemmas 16 and 17, which correspond to $r = 1$ and

Table 2: Nimbers for complete tripartite graphs with $p+q+r$ nodes.

p	q	r	nimber	rules
1	1	$x = 2y$	2	$x \geq 1$
1	1	$x = 2y-1$	1	$x \geq 1$
1	2	$x = 2y-1$	3	$x > 2$
1	3	$x = 2y$	3	$x > 3$
x	x	$y = 2w$	1	$y \geq x \ \& \ x > 1$
x	x	$y = 2w-1$	0	$y \geq x \ \& \ x > 1$
x	$x+1$	$y = 2w$	0	$y \geq x+1$
x	$x+1$	$y = 2w-1$	2	$y \geq x+1 \ \& \ x > 1$
x	$x+2$	$y = 2w$	2	$y \geq x+2 \ \& \ x > 1$
x	$x+2$	$y = 2w-1$	1	$y \geq x+2$
x	$x+z$	$y = 2w$	0	$y \geq x+z \ \& \ z = 2z' \ \& \ z > 2$
x	$x+z$	$y = 2w$	1	$y \geq x+z \ \& \ z = 2z'-1 \ \& \ z > 2$
x	$x+z$	$y = 2w-1$	1	$y \geq x+z \ \& \ z = 2z' \ \& \ z > 2$
x	$x+z$	$y = 2w-1$	0	$y \geq x+z \ \& \ z = 2z'-1 \ \& \ z > 2$

$r = 2$.

Induction step:
For any r the two possible moves are to $K_{1,r}$ and $K_{1,1,r-1}$. For the first case we know by Lemma 6 that regardless of the value of r the nimber is 0. For the second case if r is even $r-1$ is odd and so $K_{1,1,r-1}$ has nimber 1 and if r is odd, $r-1$ is even so $K_{1,1,r-1}$ has nimber 2.

Using the mex-rule we get that if r is odd $K_{1,1,r}$ has nimber 1 and if r is even $K_{1,1,r}$ has nimber 2.

□

Lemma 19. *$K_{1,2,2}$ has nimber 0*

Proof. There are two possible moves, either to $K_{2,2}$ or to $K_{1,1,2}$. The first case by Lemma 7 has nimber 1. The second case by Lemma 17 has nimber 2.

Using the mex-rule we get that $K_{1,2,2}$ has nimber 0. □

Lemma 20. *$K_{1,2,3}$ has nimber 3*

Proof. There are three possible moves. The moves result in $K_{2,3}$, $K_{1,2,2}$ or $K_{1,1,3}$. The first case by Lemma 8 has nimber 2. The second case by Lemma 19 has nimber 0. The third case by Lemma 18 has nimber 1.

Using the mex-rule we get that $K_{1,2,3}$ has nimber 3. □

Lemma 21. *Let $r \geq 2$, $K_{1,2,r}$ has nimber 0 if r is even and nimber 3 otherwise.*

Proof. We will give an inductive proof for this lemma with induction to r.
Induction hypothesis:
For every $r' < r$, $K_{1,2,r'}$ has nimber 0 if r' is even and nimber 3 otherwise.

Base Cases:
The base cases are provided by Lemmas 19 and 20, which correspond to $r = 2$ and $r = 3$.

Induction step:
There are three possible graphs to go to from $K_{1,2,r}$, namely $K_{2,r}$, $K_{1,1,r}$ and $K_{1,2,r-1}$.

Option 1. $K_{2,r}$**:** By Lemma 9 we have that if r is even the nimber is 1 and if r is odd the nimber is 2.

Option 2. $K_{1,1,r}$**:** By Lemma 18 we have that if r is even the nimber is 2 and if r is odd the nimber is 1.

Option 3. $K_{1,2,r-1}$**:** If r is even, $r-1$ is odd, so the nimber is 3 and if r is odd, $r-1$ is even, so the nimber is 0.

Using the mex-rule we get that if r is even $K_{1,2,r}$ has nimber 0 and if r is odd $K_{1,2,r}$ has nimber 3. □

Lemma 22. $K_{1,3,3}$ *has nimber* 1

Proof. There are two possible moves, either to $K_{3,3}$ or to $K_{1,2,3}$. The first case by Lemma 10 has nimber 0. The second case by Lemma 20 has nimber 3.

Using the mex-rule we get that $K_{1,3,3}$ has nimber 1. □

Lemma 23. $K_{1,3,4}$ *has nimber* 3

Proof. There are three possible moves. The moves result in $K_{3,4}$, $K_{1,3,3}$ or $K_{1,2,4}$. The first case by Lemma 11 has nimber 2. The second case by Lemma 22 has nimber 1. The third case by Lemma 21 has nimber 0.

Using the mex-rule we get that $K_{1,3,4}$ has nimber 3. □

Lemma 24. *Let $r \geq 3$, $K_{1,3,r}$ has nimber 3 if r is even and nimber 1 otherwise.*

Proof. We will give an inductive proof for this lemma with induction to r.
Induction hypothesis:
For every $r' < r$, $K_{1,2,r'}$ has nimber 3 if r' is even and nimber 1 otherwise.

Base Cases:
The base cases are provided by Lemmas 22 and 23, which correspond to $r = 3$ and $r = 4$.

Induction step:
There are three possible graphs to go to from $K_{1,3,r}$, namely $K_{3,r}$, $K_{1,2,r}$ and $K_{1,3,r-1}$.

Option 1. $K_{3,r}$**:** By Lemma 12 we have that if r is even the nimber is 2 and if r is odd the nimber is 0.

Option 2. $K_{1,2,r}$**:** By Lemma 21 we have that if r is even the nimber is 0 and if r is odd the nimber is 3.

Option 3. $K_{1,3,r-1}$**:** If r is even, $r-1$ is odd, so the nimber is 1 and if r is odd, $r-1$ is even, so the nimber is 3.

Using the mex-rule we get that if r is even $K_{1,3,r}$ has nimber 3 and if r is odd $K_{1,3,r}$ has nimber 1. □

Lemma 25. *$K_{2,2,2}$ has nimber* 1

Proof. The only graph to move to is $K_{1,2,2}$ which by Lemma 19 has nimber 0. Using the mex-rule we get that $K_{2,2,2}$ has nimber 1. □

Lemma 26. *$K_{2,2,3}$ has nimber* 0

Proof. There are two possible moves. To $K_{1,2,3}$ or to $K_{2,2,2}$. The first option by Lemma 20 has nimber 3, the second option by Lemma 25 has nimber 1.

Using the mex-rule we get that $K_{2,2,3}$ has nimber 0. □

Lemma 27. *Let $r \geq 2$, $K_{2,2,r}$ has nimber* 1 *if r is even and nimber* 0 *otherwise.*

Proof. We will give an inductive proof for this lemma with induction to r.
Induction hypothesis:
For every $r' < r$, $K_{2,2,r'}$ has nimber 1 if r' is even and nimber 0 otherwise.

Base Cases:
The base cases are provided by Lemmas 25 and 26, which correspond to $r = 2$ and $r = 3$.

Induction step:
There are two possible graphs to go to from $K_{2,2,r}$, namely $K_{1,2,r}$ and $K_{2,2,r-1}$.

Option 1. $K_{1,2,r}$**:** By Lemma 21 we have that if r is even the nimber is 0 and if r is odd the nimber is 3.

Option 2. $K_{2,2,r-1}$**:** If r is even, $r-1$ is odd, so the nimber is 0 and if r is odd, $r-1$ is even, so the nimber is 1.

Using the mex-rule we get that if r is even $K_{2,2,r}$ has nimber 1 and if r is odd $K_{2,2,r}$ has nimber 0. □

Lemma 28. *Let $p \leq q \leq r$ and $p+q+r \geq 8$ and $p+q \geq 5$ and $z > 2$. For $K_{p,q,r}$ we have the following nimbers. If $p = q$ the nimber is 1 if r is even and 0 if r is odd. If $q = p+1$ the nimber is 0 if r is even and 2 if r is odd. If $q = p+2$ the nimber is 2 if r is even and 1 if r is odd. If $q = p+z$ the nimber is 0 if $r \mod 2 = z \mod 2$ and 1 otherwise.*

Proof. We give an inductive proof for this lemma with induction to $p+q+r$.
Induction hypothesis:
Let $p+q+r = x$ and $p'+q'+r' = x'$. For every $8 \leq x' < x$, with $p'+q' \geq 5$ and $z' > 2$, $K_{p',q'}$ has the following nimber. If $p' = q'$ the nimber is 1 if r' is even and 0 if r' is odd. If $q' = p'+1$ the nimber is 0 if r' is even and 2 if r' is odd. If $q' = p'+2$ the nimber is 2 if r' is even and 1 if r' is odd. If $q' = p'+z'$ the nimber is 0 if $r' \mod 2 = z' \mod 2$ and 1 otherwise.

Base Cases:
As base cases we have $K_{1,4,4}$ and $K_{2,3,3}$. Let us prove that these cases adhere to the lemma. From $K_{1,4,4}$ we can move to $K_{4,4}$ or $K_{1,3,4}$. The first possibility by Lemma 10 has nimber 0. The second possibility by Lemma 23 has nimber 3. Applying the mex-rule we get that $K_{1,4,4}$ has nimber 1. From $K_{2,3,3}$ we can move to $K_{1,3,3}$ or $K_{2,2,3}$. The first possibility by Lemma 22 has nimber 1. The second possibility by Lemma 26 has nimber 0. Applying the mex-rule we get that $K_{2,3,3}$ has nimber 2.

Induction step:
We have multiple cases which we will handle separately.

Case 1. $p = q$ **and** r **is even:** From here the possible graphs to move to are $K_{p-1,q,r}$ and $K_{p,q,r-1}$, note that the second option is only distinct from the first option if $r > q$. For the first possibility we have that $q' = p'+1$ and r' is even, so the nimber is 0. For the second possibility we have $p' = q'$ and r' is odd which gives nimber 0. For both sub-cases $r > q$ and $r = q$ by applying the mex-rule we get that the nimber for this main case is 1.

Case 2. $p = q$ **and** r **is odd:** From here the possible graphs to move to are $K_{p-1,q,r}$ and $K_{p,q,r-1}$, note that the second option is only distinct from the first option if $r > q$. For the first possibility we have that $q' = p' + 1$ and r' is odd, so the nimber is 2. For the second possibility we have $p' = q'$ and r' is even which gives nimber 1. For both sub-cases $r > q$ and $r = q$ by applying the mex-rule we get that the nimber for this main case is 0.

Case 3. $p+1 = q$ **and** r **is even:** From here the possible graphs to move to are $K_{p-1,q,r}$, $K_{p,q-1,r}$ and $K_{p,q,r-1}$, note that the third option is only distinct from the second option if $r > q$. For the first possibility we have two sub-cases: $p = 2$ and $p > 2$. In the first sub-case the graph reached is $K_{1,3,r}$ with r is even, by Lemma 24 the nimber is 3. For the second sub-case we have that $q' = p' + 2$ and r' is even, so the nimber is 2. For the second possibility we have two sub-cases: $p = 2$ and $p > 2$. In the first sub-case the graph reached is $K_{2,2,r}$ with r is even, by Lemma 27 the nimber is 1. For the second sub-case we have that $q' = p'$ and r' is even, so the nimber is 1. Note that both possible sub-cases give nimber 1, so for the nimber of the original graph in this main case it does not matter which sub-case is true. For the third possibility we have that $q' = p' + 1$ and r' is odd, so the nimber is 2. None of the sub-cases give a possible move to a graph with nimber 0, therefore no matter what combination of sub-cases is true, by applying the mex-rule we get that the nimber for the main case is 0.

Case 4. $p+1 = q$ **and** r **is odd:** From here the possible graphs to move to are $K_{p-1,q,r}$, $K_{p,q-1,r}$ and $K_{p,q,r-1}$, note that the third option is only distinct from the second option if $r > q$.For the first possibility we have two sub-cases: $p = 2$ and $p > 2$. In the first sub-case the graph reached is $K_{1,3,r}$ with r is odd, by Lemma 24 the nimber is 1. For the second sub-case we have that $q' = p' + 2$ and r' is odd, so the nimber is 1. Note that both possible sub-cases give nimber 1, so for the nimber of the original graph in this main case it does not matter which sub-case is true. For the second possibility we have two sub-cases: $p = 2$ and $p > 2$. In the first sub-case the graph reached is $K_{2,2,r}$ with r is odd, by Lemma 27 the nimber is 0. For the second sub-case we have that $q' = p'$ and r' is odd, so the nimber is 0. Note that both possible sub-cases give nimber 1, so for the nimber of the original graph in this main case it does not matter which sub-case is true. For the third possibility we have that $q' = p' + 1$ and r' is even, so the nimber is 0. For both sub-cases $r > q$ and $r = q$ by applying the mex-rule we get that the nimber for this main case is 2.

Case 5. $p + 2 = q$ **and** r **is even:** From here the possible graphs to move to are $K_{p-1,q,r}$, $K_{p,q-1,r}$ and $K_{p,q,r-1}$, note that the third option is only distinct from

the second option if $r > q$. For the first possibility we have that $q' = p' + 3$ and r' is even, so 3 $\mod 2 \neq r' mod 2$ so the nimber is 1. For the second option we have that $q' = p' + 1$ and r' is even, so the nimber is 0. For the third possibility we have that $q' = p' + 2$ and r' is odd, so the nimber is 1. For both sub-cases $r > q$ and $r = q$ by applying the mex-rule we get that the nimber for this main case is 2.

Case 6. $p + 2 = q$ **and** r **is odd:** From here the possible graphs to move to are $K_{p-1,q,r}$, $K_{p,q-1,r}$ and $K_{p,q,r-1}$, note that the third option is only distinct from the second option if $r > q$. For the first possibility we have that $q' = p' + 3$ and r' is odd, so 3 $\mod 2 = r' mod 2$ so the nimber is 0. For the second option we have that $q' = p' + 1$ and r' is odd, so the nimber is 2. For the third possibility we have that $q' = p' + 2$ and r' is even, so the nimber is 2. For both sub-cases $r > q$ and $r = q$ by applying the mex-rule we get that the nimber for this main case is 1.

Case 7. $p + z = q$ **and** $r \mod 2 = z \mod 2$**:** From here the possible graphs to move to are $K_{p-1,q,r}$, $K_{p,q-1,r}$ and $K_{p,q,r-1}$, note that the third option is only distinct from the second option if $r > q$. For the first possibility we have two sub-cases either $p = 1$ or $p > 1$. In the first sub-case the graph reached is actually $K_{q,r}$, we have that $q \geq 4$ and $q \mod 2 \neq r \mod 2$ as $q = z + 1$, so by Lemma 15 the nimber is 1. Note that both sub-cases give nimber 1 for the second option so which sub-case is true does not effect the nimber of the original graph. In the second sub-case we have that $q' = p' + z'$ and $z' \mod 2 \neq r' mod 2$ so the nimber is 1. For the second option we have two sub-cases, either $z = 3$ or $z > 3$. If $z = 3$ we have $z' = 2$, so $q' = p' + 2$, and r' is odd so the nimber is 1. If $z > 3$ we have that $q' = p' + z'$ and $z' \mod 2 \neq r' mod 2$, so the nimber is 1. Note that both sub-cases give nimber 1 for the second option so which sub-case is true does not effect the nimber of the original graph. For the third possibility we have that $q' = p' + z'$ with $z' = z$, so $z' \mod 2 \neq r' mod 2$, so the nimber is 1. For both sub-cases $r > q$ and $r = q$ by applying the mex-rule we get that the nimber for this main case is 0.

Case 8. $p + z = q$ **and** $r \mod 2 \neq z \mod 2$**:** From here the possible graphs to move to are $K_{p-1,q,r}$, $K_{p,q-1,r}$ and $K_{p,q,r-1}$, note that the third option is only distinct from the second option if $r > q$. For the first possibility we have two sub-cases either $p = 1$ or $p > 1$. In the first sub-case the graph reached is actually $K_{q,r}$, we have that $q \geq 4$ and $q \mod 2 = r \mod 2$ as $q = z + 1$, so by Lemma 15 the nimber is 0. Note that both sub-cases give nimber 0 for the second option so which sub-case is true does not effect the nimber of the original graph. In the second sub-case we have that $q' = p' + z'$ and $z' \mod 2 = r' mod 2$ so the nimber is 0. For the second option we have two sub-cases, either $z = 3$ or $z > 3$. If $z = 3$ we have $z' = 2$, so $q' = p' + 2$, and r' is even so the nimber is 2. If $z > 3$ we have that

$q' = p' + z'$ and $z' \mod 2 = r' mod 2$, so the nimber is 0. For the third possibility we have that $q' = p' + z'$ with $z' = z$, so $z' \mod 2 = r' mod 2$, so the nimber is 0. As the first and third possible move give the same nimber, both sub-cases $r > q$ and $r = q$ are the same when applying the mex-rule. As there are no possible moves to a graph with nimber 1 both and the first possible move results in a graph with nimber 0, for the appliance of the mex-rule it does not matter whether the graph resulting from the second possible move has either nimber 0 or 2. So for all possible sub-cases, by applying the mex-rule we get that the nimber for this main case is 1. □

Theorem 5. *Given a graph $K_{p,q,r}$, the nimber for the graph is given by table 1.*

Proof. To start with for a graph to be truly denoted as a tripartite graph ($K_{p,q,r}$) the number of nodes in all three sets should be larger than 0. All values for p, q and r (in the table) are greater than 0 and values equal to 0 do not have to be in the graph.

Without loss of generality we can assume that $p \leq q \leq r$ as the sets are equivalent and do not need a specific ordering, so we can choose to assign the values of the number of nodes in the sets to p, q and r in such a way that $p \leq q \leq r$ without changing the problem. With this we have that all combinations of numbers of nodes in both sets are incorporated in the table.

The table's first and second row follow from Lemma 18.

The table's third row follows from Lemma 21.

The table's fourth row follows from Lemma 24.

The rest of the table's rows follow from Lemmas 21, 24, 27 and mainly 28. □

4.5.5 Using bipartite and tripartite tables in the algorithm for complete *k*-partite graphs

Our original algorithm for complete k-partite graphs given in Section 4.5.2 adds all integer sets for graphs $K_{1,q}$ for all q up to the maximal number of nodes in the graph with nimber 0 to the hashtable at the start of the algorithm. As by Lemma 6 we know for every q, $nb(K_{1,q}) = 0$, we can safely add all these combinations to the hashtable. These combinations then form the endpoints for the recursion in the algorithm. However with the use of the proven rules in the tables for complete bipartite and tripartite graphs we can make a more enhanced base case for the recursion. To begin with we of course can add an evaluation of the rules for complete bipartite and tripartite graphs to prevent the algorithm from going into recursion at all on these graphs; it can simply give the nimber in $O(1)$ time. More importantly we can use the rules to form new endpoints for the recursion. Just like with the combinations of $K_{1,q}$ and nimber 0 we could simply add all relevant combinations to the hashtable at the start of the algorithm. Alternatively

and less complex on execution, we could enhance the algorithm so that it goes into evaluation of the rules instead of into recursion as soon as the nimber of a complete bipartite or tripartite graph has to be evaluated. As we want the algorithm to be able to immediately evaluate complete bipartite and tripartite graphs in $O(1)$ time when they are given as the input graph, the last option is also preferable. Naturally adding all options for $K_{1,q}$ to the hashtable is no longer required when the rules are used to evaluate all complete bipartite and tripartite graphs. Furthermore note that the rules for complete bipartite graphs are never used when evaluating complete k-partite graphs with $k > 2$ as the recursion will then stop at the rules for tripartite graphs. However, we do want to add the rules for complete bipartite graphs to the algorithm, so it can evaluate these graphs when given directly as input.

4.5.6 Final thought on complete k-partite graphs

We have given a polynomial time algorithm for general complete k-partite graphs. For complete bipartite and tripartite graphs we have proven rules to determine the nimber of any such graph. These can therefore be computed in $O(1)$ time. Also we showed how to use the outcomes for complete bipartite and tripartite graphs in the general algorithm. We only proved the rules for bipartite and tripartite graphs, but this does not mean this cannot be done for greater values of k. As can be seen the number of rules and especially the complexity of the proofs increased significantly when going from bipartite to tripartite graphs. Therefore although it can be assumed that it is possible to find and prove rules for $k \geq 4$, although it will probably quickly become impractical to do so. Implementing a really vast number of rules for high values of k into the algorithm can also make the program quite complex. However when implemented correctly there should be no more problems with the vast amount of rules. If for any reason graphs with high values for k have to be evaluated, it could be useful to invest time in finding (and proving) rules for higher values of k.

5 Further use of nimbers

Knowing the nimbers for some specific graph structures is interesting, but not directly useful for finding out whether a general graph has a winning strategy for the first player or not. These more specific structures might however be useful as a stepping stone to more general graph structures, for which we might also find fast algorithms to determine the nimber of the graph. Finding such graph structures or graph characteristics is an interesting new step in research on Feedback Vertex Kayles. One interesting characteristic of graphs to start looking at is the density.

As we know that complete graphs have alternating nimber 0 and 1, this might give us what we need to quickly determine the nimber for a densely connected graph (nearly complete). Complete k-partite graphs are of course one example of a specific graph structure closely related to complete graphs for which nimbers can also be found with a fast exact algorithm. More such graph structures may still exist. On the other hand our knowledge of chains and sunflower graphs might help with finding more interesting result for sparsely connected graphs.

6 Twins and Feedback Vertex Kayles

Twins are two or more nodes in a graph that are connected to exactly the same nodes not belonging to the set of twins. We distinguish two different kinds of twins: true twins and false twins. True twins are apart from connected to the same nodes outside the collection of true twins also all connected to one another. Thus all true twins have almost exactly the same neighbour set, only differing in that they are not in their own neighbour set. False twins are only connected to all the same nodes not in the set of false twins. Thus all false twins have exactly the same neighbour set. Twins have interesting properties in Feedback Vertex Kayles. Some are equal for both kinds of twins and some differ. In this chapter we will discuss uses of twins while computing nimbers in Feedback Vertex Kayles.

6.1 Equivalence of twins

As noted in Section 3.5 isomorphic nodes have the same nimber. However as also noted that isomorphic nodes are only really interesting if we can find they are isomorphic without having to check whether the induced subgraphs from moving on such a node are isomorphic. One such case emerges when there are twins in a graph. As the twins are connected to exactly the same nodes, playing one twin is exactly equal to playing any other twin in the set. To be more precise, if we have two twins a and b in a graph g and we have one subgraph g' in which a is removed and one subgraph g'' in which b is removed the two graphs are isomorphic. We can map g'' on g' by mapping all nodes in g'' except for a on the same node in g' and mapping node a on node b in g' (if they are not already removed as well in their respective subgraphs). As a and b are connected to all the same nodes (except for each other in the case of true twins) we know that any nodes other than a that are removed from g by removing a, will also be removed from the graph by removing b, so g' and g'' contain all the same nodes except for possibly b in g' versus a in g''.

Knowing this at the very least at any step in the algorithm only one twin in each set has to be examined as a possible move, as the others will not add other nimbers to the nimber set.

6.2 True twins and Feedback Vertex Kayles

True twins are interesting in Feedback Vertex Kayles. The first intuition that they are interesting comes from the fact that a complete graph is a special case of a graph with twins, in which all nodes are twins. For complete graphs we have already proven that all even numbers of nodes have nimber 0 and all odd have nimber 1. This gives us a suspicion that maybe the amount of true twins in a graph is reducible per two twins, maintaining the same nimber. Moreover for any cycle going through a twin, a similar cycle also goes through any other twin. So as long as the number of twins remains at least two non-twins will never change from being on a cycle to not being on a cycle when twins are removed. So it would be interesting to look for a proof that true twins can be removed per two twins (to a certain minimum) or counterexamples proving this cannot always be done. However when looking into this neither a prove nor a structure for a counterexample for any number of twins could be found. The results were still rather interesting however. In the rest of this subsection we will look at a partial prove and its shortcomings, a counterexample for certain number of twins and some other results that we can conclude from this.

6.2.1 Incomplete proof for removability of true twins

Let us have a graph G with a set of k true twins and a graph H which is a copy of graph G with 2 true twins added to the set of k true twins in the graph, so H has a set of $k+2$ twins. Now if we have a winning strategy for the first player in H if and only if we have a winning strategy for the first player in G, we can conclude that removing 2 twins from a graph can always be done without changing the player with a winning strategy in the graph. There is no winning strategy for the first player if and only if there is a winning strategy for the second player. So our prove is also complete if we prove that player 1 has a winning strategy in H if player 1 has a winning strategy in G and player 2 has a winning strategy in H if player 2 has a winning strategy in G. If we have proven that player 1 has a winning strategy in H if player 1 has a winning strategy in G, the proof that player 2 has a winning strategy in H if player 2 has a winning strategy in G is almost trivial. If player 2 has a winning strategy in G, there are two options for play in H. Player 1 can play on one of the $k+2$ twins, in which case player 2 plays on another one of the twins and a graph equal to G is reached. Alternatively player 1 can play another node and create the graph H' after which we simply repeat the argument for if player 1 having a winning strategy in H' if player 1 has a winning strategy in G', where G' is the graph resulting from playing the same node as played in H to get H' in G. So we only have to prove that player 1 has a winning strategy in H if player 1 has a winning strategy in G.

Let us have a winning strategy for player 1 in G, now we use the following strategy in H. We start by playing the same node in H as would be played on G in the winning strategy for G. If in any of the following moves player 2 plays on one of the twins, player 1 also plays on a twin and a situation in the winning strategy of G is reached, so the rest of this strategy can be followed to win the game. If player 2 however refrains from playing any twins, we have two possible continuations of the game. If play continues without any twin being played, eventually only twins remain. As mentioned earlier a graph with only true twins is a complete graph and form complete graphs we know that the nimber and thus the winner is equal for any combination of k and $k + 2$ nodes, so we know that with this being a winning situation in the play starting from G, this is also a winning situation in the play starting from H. Alternatively somewhere in the winning strategy for G, the only possible move for player 1 is to move on one of the twins. In this case our strategy still follows the moves on G on H so we also move on a twin in H and keep following the same strategy.

6.2.2 Problem with the proof and counter example

The proposed strategy works when at some point in both G and H only twins remain, but by removing twins while there are still other nodes, it is possible to get into a situation where there is a direct winning move on k which is a losing move on $k + 2$. We give one example of a graph that with 4 twins has a winning strategy for player 1 but with 6 twins has a winning strategy for player 2. In Figure 13 the starting graphs for the example can be seen; on the left with 4 twins and on the right with 6 twins. In figures 14 through 20 the steps in the winning strategies and their effect on the graph are displayed. In each graph the node to be played next is marked red. In Figure 14, 16 and 18, there is a winning strategy for the player to move in the left graph. Here the node to play in a winning strategy is chosen to play in the left graph. As in all these graphs there is no winning strategy for the player to move in the graph on the right no matter which node is played, the equivalent node is chosen to play in that graph. In Figure 15 and 17, the situation is the complete opposite. In these graphs the move is chosen in the right graph and the equivalent move is played on the left graph. In Figure 19 and 20 play on the former left graph has already ended and only the play on the former right graph is displayed.

Note that for all $k > 6$ with $k \mod 2 = 0$ the nimber of the equivalent graph with k twins also is 0, this corresponds with the completion of a restricted proof in Section 6.2.4.

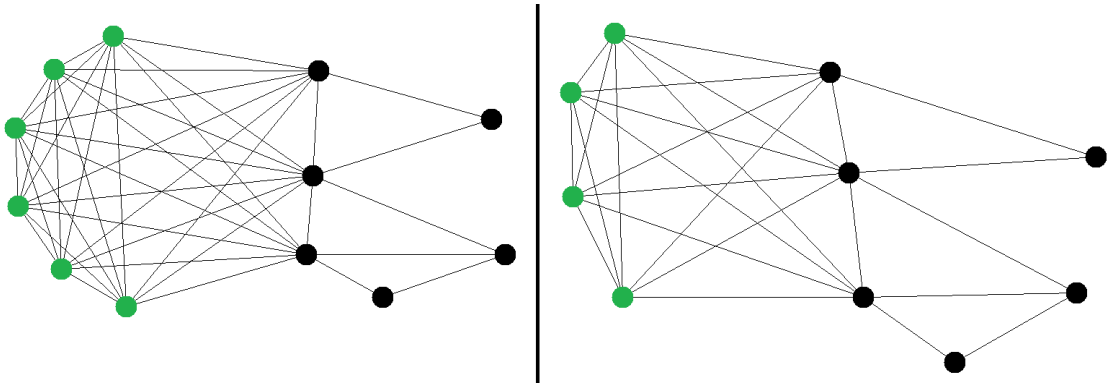

Figure 13: Examples of two graphs with 4 and 6 twins respectively but different nimbers. Green nodes are twins.

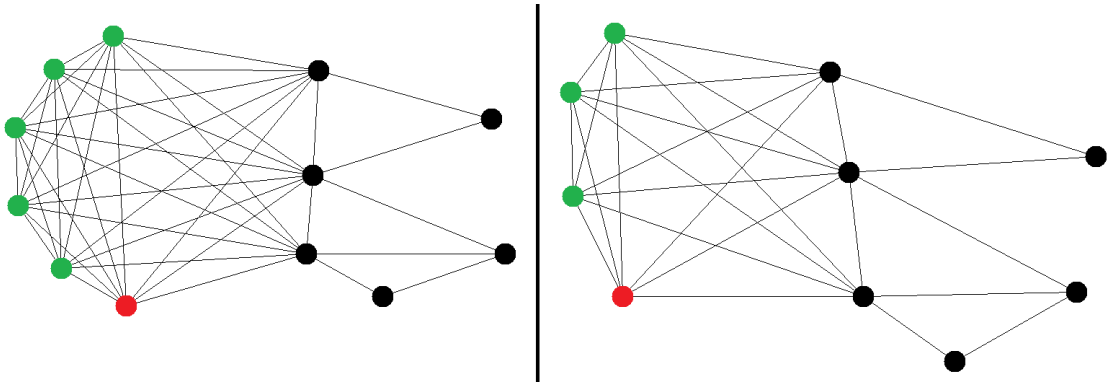

Figure 14: Step 0. Left graph has nimber 6 and right graph nimber 0. The red node is to be moved on next.

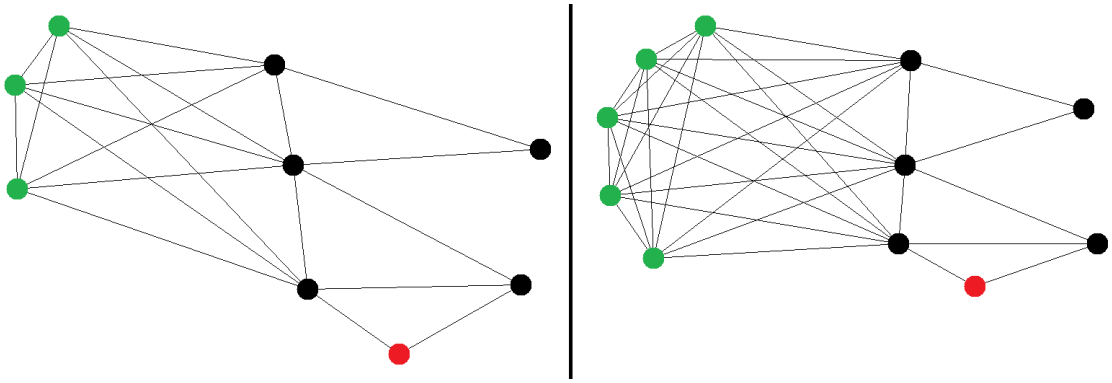

Figure 15: Step 1. Left graph has nimber 0 and right graph nimber 1. The red node is to be moved on next.

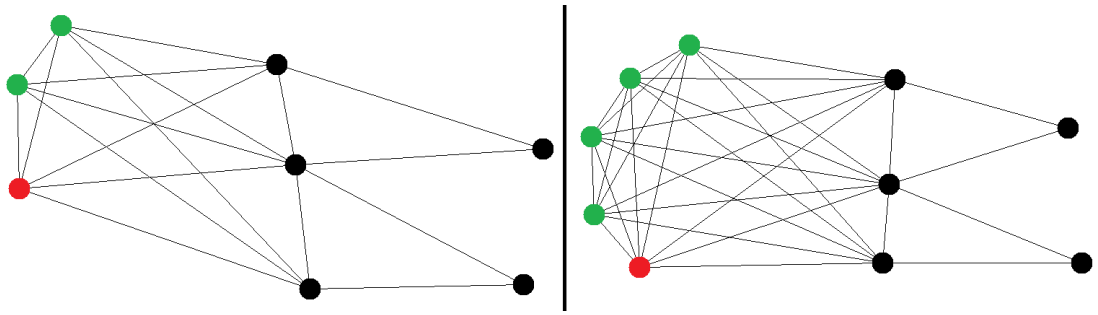

Figure 16: Step 2. Left graph has nimber 4 and right graph nimber 0. The red node is to be moved on next.

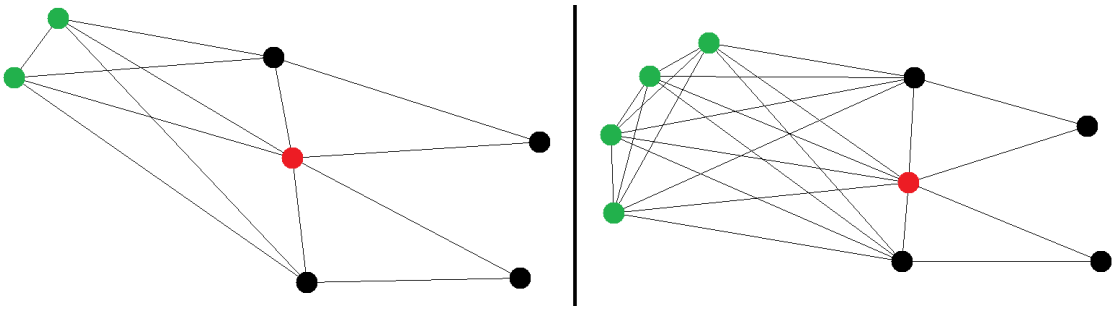

Figure 17: Step 3. Left graph has nimber 0 and right graph nimber 1. The red node is to be moved on next.

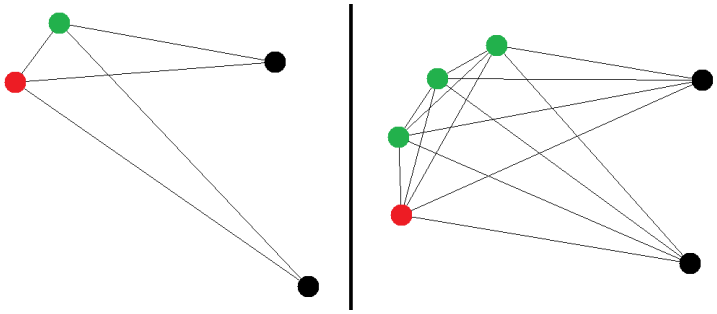

Figure 18: Step 4. Left graph has nimber 2 and right graph nimber 0. The red node is to be moved on next.

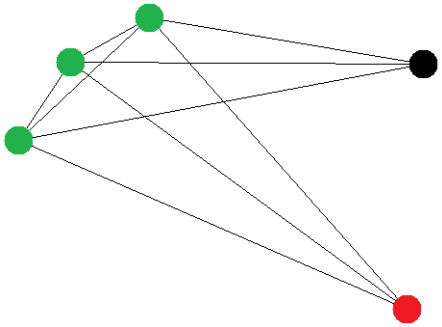

Figure 19: Step 5. This graph has nimber 1. Any non-twin node can be removed in a winning strategy.

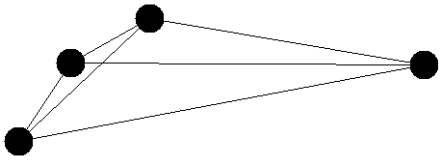

Figure 20: Step 6. Clearly whichever node is played a simple cycle of three nodes with nimber 1 remains. So this graph has nimber 0.

6.2.3 Consequences of proof at higher amount of twins

So far using a custom-made computer program to compute the nimbers of graphs with twins, only counterexamples were found for graphs with a low number of twins. As no regular pattern is found in those examples and no prove that there will be counterexamples for every number of twins, it is possible, although unlikely, that at some number of twins the proof can be completed. When the proof would be completed for a certain number of twins, we can very easily extend this proof to prove that not only the winner for the graphs with k and $k+2$ nodes are the same, but also the nimbers. We can do this by adding a new connected component to both graphs which has an equal nimber to the graph with k twins. In this way the resulting graph with k twins has nimber 0 as the nim sum of two components with the same nimber is 0. For these two new graphs our proof still holds so both must have the same nimber. This is only possible if the two components in the graph with $k+2$ twins also have the same nimber, as this is the only way for the nim sum to be 0. Which in turn means that the original graphs with k and $k+2$ twins have the same nimber.

Note that for this extension we assume that the completed proof for equal winners does not require the graph to be connected without bridges. If it only needs the graphs to be connected we can simply connect the two parts with a

bridge, as that bridge can be removed again for evaluation. It is unlikely however that the proof would need such a constraint.

6.2.4 Proof completion with restriction on number of non-twins.

Let us denote the number of nodes in the biggest group of twins in graph G as $nt(G)$ and the number of other nodes (non-twins) as $nn(G)$. We can prove the following lemma.

Lemma 29. *Let G_k be a graph with $nt(G) = k$ and G_{k+2} be a graph with $nt(G) = k + 2$. If $nt(G_k) \geq nn(G_k)$ there is a winning strategy for player 1 in G_k if and only if there is a winning strategy for player 1 in G_{k+2}.*

Proof. The first part of the prove is given in Section 6.2.1. The proof is finished as follows.

If player 2 never moves on a true twin, eventually, as there are at least as many true twin nodes as non-twin nodes and at most one in two nodes removed is a true twin, all non-twin nodes will be removed from the graph and only true twins remain. As established before a group of true twins without other nodes is a complete graph. Let k' be the remaining number of true twins in G_k after the strategy being applied to it. By Lemmas 4 and 5 we have that the nimbers for k' and k are the same, so both resulting graphs have the same winner. □

6.2.5 Conclusion on twins

Although twins are interesting and their potential for Feedback Vertex Kayles seems high, it appears not possible to give a kernel on the number of twins. Although no strict rules for counterexamples have been found it is very likely that for every n a counterexample can be found. When only one graph is examined we can however decrease the number of twins by 2 as long as it stays higher than the number of other nodes without changing the winner.

7 Complexity of the problem

Apart from nimbers and fast algorithms for specific cases, an interesting open problem, is the complexity of Feedback Vertex Kayles. It is expected that this problem is PSPACE-complete; however a proof is still to be found. There are several facts that strengthen the suspicion that this problem is PSPACE-complete. To start with the game is closely related to the Feedback Vertex Set problem, in which a minimal set of vertices has to be found which when removed from the graph leaves a graph with no cycles. This problem is NP complete, it seems unlikely that

Feedback Vertex Kayles would be an easier problem than Feedback Vertex Set, as the minimal Feedback Vertex Set would probably be one of the considered play sequences in Feedback Vertex Kayles. Furthermore Feedback Vertex Kayles seems to be similar to other PSPACE-complete games like Kayles, it even seems harder than Kayles, as a move generally removes less structure from the graph than with Kayles. This last remark however also makes it harder to find a similar proof for the complexity of Feedback Vertex Kayles as for the complexity of Kayles. The latter proof adds a lot of nodes and edges, which ensure that if an "illegal" move is made, the game is lost because a lot of nodes get removed at once. In Feedback Vertex Kayles however adding more nodes and/or edges does not increase the number of removed nodes for a chosen node and thus only creates a more complex structure.

Conjecture 1. *Feedback Vertex Kayles is PSPACE-complete.*

We must note that if our conjecture turns out to be false and Feedback Vertex Kayles is proven to be in P, polynomial time algorithms for specific cases become a lot less interesting.

8 A fast exact algorithm for general graphs

Assuming Feedback Vertex Kayles is indeed PSPACE-complete, we know that an algorithm for Feedback Vertex Kayles on general graphs can never be faster than exponential. However it is interesting to find an algorithm with an upper bound time complexity that is as low as possible. A first idea might be to use nimbers and compute an upper bound just like done for Kayles in [Bodlaender et al., 2015]. The upper bound given by this paper relies on the fact that not all combinations of moves are possible, as removing one node from the graph also removes all its neighbours. For Feedback Vertex Kayles we of course have that removing a node effectively also removes all nodes that were only on a cycle that is removed by removing the first node. However there is no guaranty as to which nodes are removed. Moreover in a complete graph, even though we have proven this to be an easily solvable case, all but two nodes have to be chosen before the game ends, as each node remains on at least one cycle. This makes it impossible to simply use the simple algorithm given for Kayles, for Feedback Vertex Kayles and compute a similar upper bound. However this does not necessarily mean that the upper bound cannot get lower than 2^n. The example we give for a case in which all nodes have to be played is a complete graph. Although in such a graph the game continues until all but two of the nodes are played, we have proven that nimbers for complete graphs alternate between 0 and 1 for even and odd numbers of nodes respectively and thus computing the nimber (or winner) for such a graph takes

only $O(1)$ time. Furthermore if, when evaluating any other graph, the resulting subgraph after any number of moves is a complete graph, the nimber of that subgraph can be computed in $O(1)$ time. So this can limit the number of possible subgraphs and may lead towards a better upper bound for the algorithm. For a good upper bound we should combine different simpler structures for which we have found the nimber easy to compute.

9 Future Work

Although some interesting result are described in this thesis there is still much more that could be researched. Also some results might be further evaluated to find more useful implications.

To start with of course the complexity of Feedback Vertex Kayles is an interesting field for future work. Although we conjectured that Feedback Vertex Kayles is PSPACE-complete, no attempts at a prove have been successful. Proving the complexity of Feedback Vertex Kayles thus stays an open problem. Solving this problem is also important to validate (or possibly invalidate) the significance of results on specific graph structures.

In this thesis a number of specific graph structures and specific node types in graphs and their relation to Feedback Vertex Kayles are discussed. There could however be (far) more graph structures or other specifications that are interesting for Feedback Vertex Kayles. Some suggestions are already given like looking at densely or sparsely connected graphs. Also graph structures with a low bound on the maximal nimber as discussed in Section 3.6 might be interesting. For graph structures with a low maximal nimber it is also plausible that the structure allows for simpler evaluation than evaluating all subgraphs.

The general algorithm that is discussed in section 8 could possibly be improved upon. Using the insights from this thesis and possible future insights it might be possible to insert polynomial time sub-routines into the algorithm that result in a better expected time bound. It might even be possible for some combination of structures to prove that any graph will end in such a structure at a significant point somewhere during the recursive evaluation. If for all these graph structures a fast algorithm is known this can reduce the bound for the general algorithm.

Finally the upper bound (and possibly the lower bound) for the general algorithm might be improved. An analysis as done for Kayles by [Bodlaender et al., 2015] would be interesting. As discussed in Section 8 a similar analysis does not benefit Feedback Vertex Kayles. A different analysis, possibly using insight on specific graph structures might however exist. Finding and performing such an analysis is an interesting option for future work.

10 Conclusion

Feedback Vertex Kayles is an interesting game and the problem to find a winning strategy in the game is even more interesting. There are good reasons to assume it is PSPACE-complete, but so far no proof has been provided.

In the general algorithm using nimbers, it is determined recursively whether for a graph there is a wining strategy for the first player or not. Although all possible move sequences can be important for the solution to this problem the algorithm does not have to evaluate all $O(n!)$ possibilities. Using memoisation the computation time can be reduced to $O(2^n)$.

The workings of the game allow for Feedback Vertex Kayles to be more easily evaluated on specific graph structures. In this thesis we have explored chains, sunflower graphs, complete graphs and complete k-partite graphs as specific graph structures. When the graph is known to adhere to one of these structures for all but complete k-partite graphs Feedback Vertex Kayles can be evaluated in constant time. For the evaluation of general complete k-partite graphs we have given a polynomial time algorithm. For $k = 2$ and $k = 3$ we have given tables so that these cases can also be evaluated in constant time when the number of nodes per independent set is known. It is interesting that these structures can be solved with fast algorithms especially when Feedback Vertex Kayles is indeed proved to be PSPACE-complete. The structures may also be used to enhance an algorithm exploring whether there is a winning strategy for general graphs.

Another interesting phenomena for Feedback Vertex Kayles are twins. These nodes seemed extremely promising for Feedback Vertex Kayles. Although we have shown that they cannot always be removed per two nodes without changing the nimber, they are still interesting in the problem. As all twins are isomorphic nodes and we have argued that for all isomorphic nodes only one node has to be evaluated, they at least provide a way to decrease the amount of evaluation an algorithm has to do.

In conclusion Feedback Vertex Kayles is an interesting and complex problem. It is hard for the general graphs but has enough interesting characteristics to be solved with faster algorithms in specific cases.

References

E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*. Academic Press, London, 1982. ISBN 0-12-091150-7.

Hans L. Bodlaender and Dieter Kratsch. Kayles and nimbers. *Journal of Algorithms*, 43:106–119, 2002. doi: 10.1006/jagm.2002.1215.

Hans L. Bodlaender, Dieter Kratsch, and Sjoerd T. Timmer. Exact algorithms for Kayles. *Theoretical Computer Science*, 562:165–176, 2015. doi: 10.1016/j.tcs.2014.09.042. URL `www.elsevier.com/locate/tcs`.

John H. Conway. *On Numbers and Games*. IMA, 1976.

Aviezri S. Fraenkel. Combinatorial games: selected bibliography with a succinct gourmet introduction. 1996.

Patrick M. Grundy. Mathematics and games. *Eureka*, 2(5):6–8, 1939.

Richard K. Guy and Cedric A. B. Smith. Mathematical Proceedings of the Cambridge Philosophical Society The G-values of various games. *Mathematical Proceedings of the Cambridge Philosophical Society*, 52(52):514–526, 1956. doi: 10.1017/. URL `http://journals.cambridge.org/PSPhttp://journals.cambridge.org/abstract{_}S0305004100031509`.

Melissa Huggan. Impartial Intersection Restriction Games. Master's thesis, Carleton University, 2015. Canada.

Simon Prins. Finding a winning strategy in variations of Kayles. Master's thesis, Utrecht University, 2015. The Netherlands.

Thomas J. Schaefer. Complexity of Decision Problems based on Finite Two-Person Perfect-Information Games. In *Eighth annual ACM Symposium on Theory of Computing*, pages 41–49, New York, 1976. doi: 10.1145/800113.803629.

Thomas J. Schaefer. On the Complexity of Some Two-Person Perfect-Information Games. *Journal of Computer and System Sciences*, 16:185–225, 1978.

Richard Sprague. Uber mathematische kampfspiele. *Tôhoku Math. J*, 41(438-444): 6, 1935.

R. Endre Tarjan. A note on finding the bridges of a graph. *Information Processing Letters*, 1974. ISSN 00200190. doi: 10.1016/0020-0190(74)90003-9.