Impartial Intersection Restriction Games

by

Melissa Huggan

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Science

in

Pure Mathematics

Carleton University
Ottawa, Ontario

Abstract

Intersection restriction games are games played on hypergraphs in which options for a player are restricted based on previous play via some intersection property. This paper focuses on two games within this class: Arc-Kayles and a Triple Packing game. Arc-Kayles is a game where, on their turn, players remove an edge and all adjacent edges from a graph. Together, players are forming a maximal matching. The Triple Packing game is a combinatorial design game where players are choosing triples such that no two triples chosen share a pair. Both games are played under normal play. We give new results for Arc-Kayles played on a special star graph and the wheel graph as well as partial results for the Triple Packing game played on the complete graph.

Acknowledgements

I would like to thank my supervisor Dr. Brett Stevens. Brett's knowledge and insight into mathematical connections has been very inspirational to my academic curiosity. Brett has provided me with invaluable national and international academic opportunities which expanded my mathematical thinking to new creative levels.

I would also like to thank my examination committee members for their suggestions and thought-provoking questions throughout my thesis defence. My committee members include: Dr. Kevin Cheung from the School of Mathematics and Statistics at Carleton University; Dr. Jason Etele from the Department of Mechanical and Aerospace Engineering at Carleton University; and Dr. Lucia Moura from the School of Electrical Engineering and Computer Science at the University of Ottawa.

Lastly, I would like to thank my family for being there for me throughout this amazing adventure.

Contents

Chapter 1

Combinatorial Game Theory

1.1 Introduction

Unless otherwise stated, all definitions from this section are from [1].

A *combinatorial game* is a two player, perfect information game, involving no element of chance, where players alternate turns. The players are opponents, and all information is available to both players about past play and all current available options (moves). Combinatorial games discussed here are finite, and the end condition is considered to be under *normal play*, where the last player to move wins. Alternatively, it would be called *misère play* where the last player to move loses.

There are several different ways to classify games. One way to classify a game is by the options available to a player on their turn.

Definition 1. A game is called *impartial* if both players have the same set of options available to them at all times throughout the game. Otherwise, the game is called *partizan*.

For partizan games, we need two different names for the players. Traditionally, they are called the Right player and the Left player (named after an important researcher who established many of the foundations of Combinatorial Game Theory, Richard Guy, and his wife, Louise),

Class	Description
$\mathcal{N}$	The Next (first) player can force a win.
$\mathcal{P}$	The Previous (second) player can force a win.
$\mathcal{L}$	The Left player can force a win regardless of who moves first.
$\mathcal{R}$	The Right player can force a win regardless of who moves first.

Table 1.1: Outcome class descriptions [1].

where each have a different set of allowable moves on their turn. In game classification, there are also outcome classes. *Outcome classes* describe who has a winning move in the game. Table 1.1 gives a description of the outcome classes. For partizan games, there are four outcome classes: Next (first) player win, $\mathcal{N}$, Previous (second) player win, $\mathcal{P}$, Left player win, $\mathcal{L}$, or Right player win, $\mathcal{R}$. For impartial games there are only two outcome classes. This is summarized in the following theorem.

Theorem 1. [1] *If $\mathcal{G}$ is an impartial game, then $\mathcal{G}$ is either in $\mathcal{N}$ (next player win) or $\mathcal{P}$ (previous player win).*

Proof. Suppose there is a way for the Left player to win going first. If Right were to move first, they could use Left's strategy and win the game since $\mathcal{G}$ is impartial, and so both Left and Right have the same options. The argument is symmetric for Right having a winning move going first, and Left adopting Right's strategy. This is called a strategy-stealing argument and thus, if $\mathcal{G}$ is impartial, then $\mathcal{G}$ is either in $\mathcal{N}$ or $\mathcal{P}$. □

Throughout this paper, we will only be examining impartial games. Hence, the partizan counterpart of combinatorial game theory will not be mentioned hereafter.

All impartial games have values, called the Grundy value [1]. The value of a game will determine the outcome class. This means that we need a method to calculate the game value in order to know who will win from any position in the game. The value of an impartial game is calculated using a function called the minimum excluded value.

Definition 2. The *minimum excluded value* (mex) of a set S is the smallest non-negative integer not appearing in S.

Example 1. *Let* $S = \{1, 2, 4\}$. *Then* $mex(S) = 0$.

Example 2. *Let* $S = \{0, 2, 3, 5\}$. *Then* $mex(S) = 1$.

In order to better understand how this definition is used to determine the value of a game, we define the game Nim which is the basis for most impartial combinatorial game theory. Nim was fully solved by Charles L. Bouton in 1901 [8].

Game 1. *Nim is a game played on heaps of counters, where the heap sizes are finite. On their turn, players may remove any positive integer* k *of counters from any single heap of size* n, *where* $1 \leq k \leq n$. *Players must remove at least one counter on their turn. The last player to move wins.*

In a game of Nim, if there is only one non-empty heap of counters, the first player always has a winning move. Their strategy is to remove the entire heap. If the current (next) player does not have a move (the heap is empty), the game position is a second (previous) player win and has value 0. Note that the title of Next and Previous alternates between players as the players alternate turns. For example, suppose there are two players (Player A and Player B) playing Nim on a heap of size 2 and Player A goes first. Player A is considered to be the Next player for the instance of the game Nim(2). Suppose Player A takes away all of the counters from the heap leaving the empty heap for Player B. Then, during the second turn (the turn of Player B), Player A is considered to be the Previous player in the sub-game Nim(0).

The options of a game are all possible moves players have available to them. The value of a game is defined by its options [1]. Considering the game tree of all possible moves, the value at any node in the tree is calculated from the minimum excluded value of the values of

its children. We need notation to capture the essence of the mex function. Playing a game of Nim on a heap of size $n > 0$, a player has every possible non-negative integer $i < n$, heap size as an option. For instance, the game with a heap of size 0 means that there is no option for the next player and hence is a previous player win. The notation for this is as follows: $|\mathcal{G}| = 0$, with options $\{\} = 0$, the mex of $\{\}$ is 0. For a heap of size 1, the options are to leave an empty heap which gives the following: $\{0\} = *1$. The 1 is the mex of the options, $\{0\}$. This process continues for any $n \in \mathbb{Z}^{\geq 0}$. These values are called *nimbers* and are formally denoted by $*n$, $n \in \mathbb{Z}^{\geq 0}$. In practice, when discussing only impartial games, we omit the $*$.

If the context is clear we write $\mathscr{G}(n)$, meaning a game value (Grundy value) for the game of size equivalent to n. Otherwise we write $\mathscr{G}(GameName(n))$, indicating the game played (GameName) as well as the heap size equivalence. For any impartial game which has non-negative integer parameters we can define the *nim-sequence* as the nim-value (Grundy value) when all but one parameter is fixed and we index the sequence by the varying parameter. Thus the nim-sequence for the game Nim with only one heap of size n as n goes to infinity is $\mathscr{G}(n) = n$, for all $n \geq 0$. The nim-sequence definition generalizes as follows to other impartial games.

Definition 3. Let $\mathscr{G}(n)$ denote the value of a game with a varying parameter, n. Then the *nim-sequence* for the game is $\mathscr{G}(0), \mathscr{G}(1), \mathscr{G}(2) \ldots$.

Let's now consider an example. The game tree of a Nim game with heap size 3, denoted Nim(3), is pictured in Figure 1.1. The game tree starts without any values assigned to any level of the tree. Starting with the root node (Nim(3)), we break down each position into their allowable options. For example, at the root, Nim(3), the current player can either remove all the counters (which leaves the empty heap), remove one counter (which leaves a heap of size 2) or remove two counters (which leaves a heap of size 1). Thereafter, this process is repeated for its non-empty sub-positions until each path from the root to an end-game (leaf of the game tree) is empty. Now we assign values as follows. All the leaves (empty

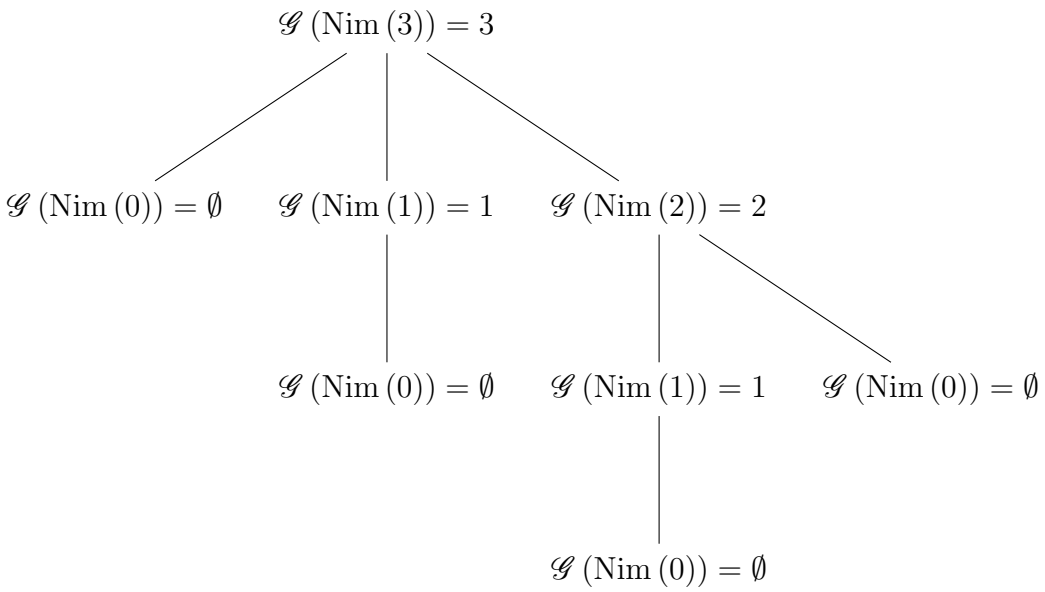

Figure 1.1: Game tree calculation for the value of Nim(3).

heaps) are assigned the value 0 which signifies that neither player has an option (and hence this sub-game is a second player win). Then we backtrack up the levels of the tree, taking the mex of the children nodes at each step. For example, when we calculate the value of the whole game, we look at the values of the children of that node, $\{0, 1, 2\}$. The mex of this set is 3 and hence is the value of the game Nim(3). It can easily be seen that as the heap size grows the number of computations (and size of the game tree) could grow exponentially.

More commonly we play Nim with several heaps of counters. Playing this game, is slightly different from Nim on a single heap. On their turn, players may remove any positive number, i, of counters from a single heap of size m, where $1 \leq i \leq m$. We call this game a disjunctive sum of single heaps. Arithmetic works a little bit differently when considering the sum of two (or more) games. To compute the disjunctive sum of a set of impartial games we sum the values of each disconnected sub-game, in binary, without carrying. This is more commonly known as the XOR (exclusive or) function, and is represented as $\oplus$. Throughout this paper, we slightly abuse this notation and also consider $\oplus$ to mean the game play on

two (or more) sub-games, $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$, of an overall game $\mathcal{H}$, as $\mathcal{H} = \mathcal{G}_1 \oplus \mathcal{G}_2 \oplus \cdots \oplus \mathcal{G}_n$.

Impartial games are studied by determining winning positions for first (next) players and second (previous) players. We call a game a *P-position* (*N-position*) if the previous (next) player can always win from that game position. Arguably the most important theorem in impartial game theory was developed by Bouton [8] and later formalized as follows:

Theorem 2. [1] *Let $\mathcal{G} = Nim\,(a, b, c, \ldots, k)$. Then $\mathcal{G}$ is a P-position if and only if $a \oplus b \oplus \cdots \oplus k = 0$.*

Proof. Suppose $a \oplus b \oplus \cdots \oplus k = 0$. Then, using induction on the height of the game tree, we need to show that every move is to a non-zero position, which by induction is winning for the subsequent player. Suppose without loss of generality that the next player chooses a heap of size a and removes r counters, where $a \geq r > 0$. Then $(a - r) \oplus b \oplus \cdots \oplus k \neq a \oplus b \oplus \cdots \oplus k = 0$.

Conversely, we need to show that if $a \oplus b \oplus \cdots \oplus k \neq 0$ then we can find a move which brings the game to zero. Let $q = a \oplus b \oplus \cdots \oplus k$. Then $q = q_0 q_1 \cdots q_k$ where q_i is a 0 or 1 based on the binary representation of q. Since the XOR sum is not equal to zero, q has a leftmost position with a 1, say q_j. Since there is a 1 in position q_j there must be a heap from $(a, b, \ldots, k)$ with a 1 in the same position. Without loss of generality suppose a had a 1 in position a_j. Since $q \oplus a < a$, we can find a $d < a$ such that $a - d = a \oplus q$ and so we remove d counters from a. So, $(a - d) \oplus b \oplus \cdots \oplus k = (a \oplus q) \oplus b \oplus \cdots \oplus k = a \oplus (a \oplus b \oplus \cdots \oplus k) \oplus b \oplus \cdots \oplus k = (a \oplus a) \oplus (b \oplus b) \oplus \cdots (k \oplus k) = 0$. Hence there is a move which brings the game to a zero sum. □

Theorem 2 holds true for any impartial game $\mathcal{G}$, by induction on the height of the game tree, as the values for an impartial game are, by definition, determined by the mex function. We restate the theorem in general, as a corollary.

Corollary 1. *Let $\mathcal{G}\,(a, b, c, \cdots, k)$ be a game with k components played as a disjunctive sum of components. Then $\mathcal{G}$ is a P-position if and only if $\mathscr{G}\,(\mathcal{G}\,(a)) \oplus \mathscr{G}\,(\mathcal{G}\,(b)) \oplus \cdots \oplus \mathscr{G}\,(\mathcal{G}\,(k)) = 0$.*

Example 3. *Let* $\mathcal{G} = Nim\,(3, 6)$. *The binary expansions for* 3 *and* 6 *are not the same; their nim-sum is* 5. *This means by Theorem 2 that there is a winning move for the next player. The leftmost bit of* 5 *is in the third position;* 6 *also has a bit in this position. We find that* $6-3 = 6 \oplus 5$. *The next players' winning strategy is to reduce the heap of size* 6 *by* 3, *bringing the game value to* 0. *This results in two heaps of equal size and thus allows the first player to win by mimicking whatever the second player chooses to do in the opposite heap of size* 3. *This is called the Tweedledum-Tweedledee strategy.*

Once we have a nim-sequence, we look for regularities. If patterns are found, we may have a clear way to identify who will win based off of a finite number of values from the nim-sequence. For the games we are considering here, the *periodicity* of a sequence is important.

Definition 4. A nim-sequence is said to be *periodic* if there is some $l \geq 0$, $p > 0$ so that $\mathscr{G}(n+p) = \mathscr{G}(n)$ for all $n \geq l$. And $\mathscr{G}(0), \mathscr{G}(1), \ldots, \mathscr{G}(l-1)$ is the pre-period of the sequence.

Before examining games, we first prove a lemma on the boundedness of nim values.

Lemma 1. [4] *In Nim*$(n_1, \cdots, n_m)$ *where* $1 \leq i \leq m$, $\mathscr{G}(\oplus n_i) \leq \sum n_i$.

Proof. We will prove the lemma by induction. Suppose we have one heap of size n. Recall in Nim, $\mathscr{G}(n) = n \leq n$. Now, suppose we have two heaps of size m and n and both m and n have a 1 in position j. Then $\mathscr{G}(m \oplus n) = \mathscr{G}((m - 2^j) \oplus (n - 2^j))$ because as we add columnwise in binary, the common values summing to zero is equivalent to subtracting out the number of common positions (j) for each heap value in binary. And by induction

$$\mathscr{G}(m \oplus n) = \mathscr{G}((m - 2^j) \oplus (n - 2^j)) \leq m + n - 2^{j+1} \leq m + n.$$

If m and n have no common position with a 1, $\mathscr{G}(m \oplus n) = m + n \leq m + n$. As this is an iterative process, it is sufficient to consider only two heaps and this can be extended to k heaps, where $k \geq 3$ by grouping. □

n	options	values	mex	$\mathscr{G}(Nim(n))$
1	$\oslash$	0	1	1
2	$\oslash$, {1}	0, 1	2	2
3	$\oslash$, {1}, {2}, {1, 1}	0, 1, 2	3	3
4	$\oslash$, {1}, {2}, {3}, {1, 1}, {1, 2}	0, 1, 2, 3	4	4
5	$\oslash$, {1}, {2}, {3}, {4}, {1, 1}, {1, 2}, {1, 3}, {2, 2}	0, 1, 2, 3, 4	5	5

Table 1.2: Double-Split Nim base cases.

This shows that as we add games in disjunctive sums, the value of $\mathcal{G}$ is bounded above by the sum of nimbers. This is particularly important when we consider the game *Split Nim*. This is the game Nim where players are given an additional option after a counter (or several) has been removed to split the heap into two (or possibly more) smaller heaps. Nim variants have been a popular topic of study for modern combinatorial game theory. Changing the rules of the game can sometimes lead to the same nim-sequence and this tells us that the games are equal. Formally, two games $\mathcal{G}$ and $\mathcal{H}$ are *equal* if for all games $\mathcal{X}$, $\mathcal{G} \oplus \mathcal{X}$ has the same outcome as $\mathcal{H} \oplus \mathcal{X}$. We now examine two versions of Split-Nim and show that, as games, they are equal to Nim.

Game 2. *Double-Split Nim: On your turn you must remove a positive number of counters from any single heap and then, if you would like to, split the heap into two smaller heaps. The last player to move wins.*

Proposition 1. *Double-Split Nim is equal to Nim.*

Proof. In Table 1.2, we see the values for small Nim heaps (our base cases). Suppose $n = k$ and $\mathscr{G}(k) = k$. Now we consider the case when $n = k + 1$. We observe that we must remove at least one counter from a heap and then, if we want, split the heap into two smaller heaps. If we do not split after reducing the heap to size i then $\mathscr{G}(i) = i$, by induction. Thus $\mathscr{G}(n)$ has i as an option for all $i < n$. If a player decides to split into two heaps of size s and t, after removal of i counters, then by Lemma 1, we know that the game values do not increase

beyond $s + t = n - i < n$. This means we do not get new values contributing to the mex calculation. And we conclude that $\mathscr{G}(n) = mex(0, 1, 2, \ldots, k) = k + 1$. □

If players are permitted to split into more than two piles after counter removal (Multi-Split Nim) the use of Lemma 1 is similar and we get the game is still equal to Nim.

Proposition 2. *Multi-Split Nim is equal to Nim.*

This shows that Nim, Double-Split Nim and Multi-Split Nim all have the same nim-sequence and thus are all equal as games.

The nim-sequence can however change dramatically depending on the conditions of the game. *Grundy's game* is a splitting game where players may choose a heap and split it into unequal sized heaps. This game remains unsolved. It is assumed to be periodic [4] but the period is unknown and likely to be extremely large.

One interesting way to study games is by modelling games on structures which are well studied. This allows the study of game outcomes via the well known characteristics of its underlying structure. One such structure is a graph. We now present some graph theory background in order to facilitate discussion of games to come.

1.2 Graph Theory

The definitions from the following section are from [7] unless otherwise stated. A *graph* G is an ordered pair $(V(G), E(G))$ consisting of a set of vertices, $V(G)$, and a set of edges, $E(G)$, together with an incidence function ψ_G that associates with each edge of G an unordered pair of vertices of G. A *neighbour* of a vertex v, in a graph G, is a vertex u, distinct from v, which is connected to v by an edge, e. The *degree* of a vertex v in a graph G, denoted by $d_G(v)$, is the number of edges of G incident with v, each loop counting as two edges.

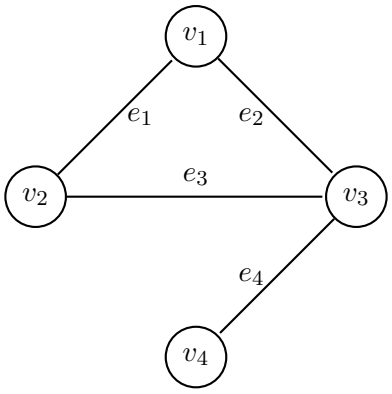

Figure 1.2: Example of a graph G.

Example 4. *Let $G = (V(G), E(G))$ from Figure 1.2. The vertices of G are defined as $V(G) = (v_1, v_2, v_3, v_4)$. The edges of G are defined as $E(G) = (e_1, e_2, e_3, e_4)$. The incidence function ψ_G is defined as $\psi_G(e_1) = \{v_1, v_2\}$, $\psi_G(e_2) = \{v_1, v_3\}$, $\psi_G(e_3) = \{v_2, v_3\}$, $\psi_G(e_4) = \{v_3, v_4\}$.*

A graph F is called a *subgraph* of a graph G if $V(F) \subseteq V(G), E(F) \subseteq E(G)$, and ψ_F is the restriction of ψ_G to $E(F)$.

There are a multitude of properties to analyze in a graph. Two properties of particular importance for work later presented in this paper are independent sets and maximal matchings. An *independent set* in a graph G is a set of vertices, V', which are pairwise non-adjacent. V' is *maximal* if no other vertex, v of G such that $v \notin V'$, can be added to V' while maintaining pairwise non-adjacency. The set V' is *maximum* if its size is the largest of all independent sets of G. The analogue for edges is referred to as a matching. A *matching* M in a graph G is a set of edges such that no two edges in M share a vertex. A *maximal matching* is a matching which cannot be extended. A *maximum matching* is a maximal matching of largest size for the graph G.

1.3 Games on Graphs

If we consider games and graphs together, we get a whole new area of mathematics to explore. Many games can be modelled on a graph and then we can use the properties of a graph to examine the sub-positions of the game.

Masahiko Fukuyama created a game called *Nim on graphs* [19]. Nim on graphs is set up as follows: Fix a graph and assign a positive integer weight to every edge of the graph. Choose a vertex as the starting vertex and identify it with a playing piece. A player then chooses an edge incident to the starting vertex. The edge must have a strictly positive weight assigned to it. The player decreases the weight of that edge by any positive integer, then slides the playing piece to an adjacent vertex via the edge used. This continues, alternating players, until a player can no longer move. The last player to move wins.

Nim is an example of *Nim on graphs*, when we consider the graph on two vertices with multiple edges, where the edges are assigned the number of counters in each heap. Choosing an edge in *Nim on graphs* is the same as choosing a heap in Nim; reducing a weight is the same as removing counters from a heap, and switching the playing piece between vertices is just the act of taking turns between players. This is just one example of a graph game from the literature; for other examples, see [13], [21].

Here, as a starting point, we first define a few simple games and then discuss some results.

Game 3. *Graph and its connected components (GCC): On your turn you may remove any set of edges from a connected component of G. The last player to move wins.*

This is part of folklore but for completeness we include a proof.

Lemma 2. *Let G be a connected graph with m edges and $m' < m$. Then there exists a connected subgraph H with exactly m' edges.*

Proof. Consider a connected graph G with m edges. We need to show that for every $m' < m$ there exists a subgraph of G which is connected and has m' edges.

We begin by removing edges incident to leaves one at a time. If there is an option to remove an edge incident to a leaf, that always takes priority to deleting any other edge. If there are no leaves, we delete an edge from a cycle. This keeps the graph connected. Recursively applying this algorithm will account for all values of $m' < m$. □

Proposition 3. *GCC is equal to Nim if the graph of G is connected.*

Proof. Proof by (top down) induction. Base cases: Suppose G has zero edges. Then the options of G are empty, and $\text{mex}(\emptyset) = 0$ so the value of G is 0. If G has one edge, then G can either have one vertex and a loop or two vertices and one edge between them. If we delete that edge, then we are taking the minimum excluded value of $\{0\}$ which equals 1. Hence the value of G is 1.

Induction hypothesis: If G has n edges, then the value of G is n.

Induction step: Consider the game on H which has $n+1$ edges. By Lemma 2 there exists a graph $H' \subset H$ such that the number of edges of H' is $m' < n+1$ and H' is connected so nimber m' is an option of H for all $m' < n+1$.

Let S be a subset of edges of H and $H' = H \backslash S$. Suppose we have components $H_i, 1 \leq i \leq t$ with m_i edges each. By induction hypothesis and by $\mathscr{G}(H') = \oplus \mathscr{G}(H_i) = \oplus m_i \leq \sum m_i < n+1$, the values of moves on H are $\{0, 1, \ldots, n\}$ so $\mathscr{G}(H) = mex(0, 1, \ldots, n) = n+1$

□

The merging of graph theory and game theory is the crux of the analysis of the games we study in this paper. Modelling a game using a graph structure permits a nice construction when writing algorithms to solve the game, especially as the number of vertices gets large. Adapting the nim-sequence definition, a sequence $S = s_0 s_1 s_2 \ldots s_n \ldots$ denotes the nim-sequence for a graph class of a particular game, where (unless otherwise stated) each s_i represents the value of that game on i vertices [3].

	t→	0	1	2	3	4	5	6	7	8	9	10	11
s↓													
0		0	1	2	3	1	4	3	2	1	4	2	6
12		4	1	2	7	1	4	3	2	1	4	6	7
24		4	1	2	8	5	4	7	2	1	8	6	7
36		4	1	2	3	1	4	7	2	1	8	2	7
48		4	1	2	8	1	4	7	2	1	4	2	7
60		4	1	2	8	1	4	7	2	1	8	6	**7**
72		**4**	**1**	**2**	**8**	**1**	**4**	**7**	**2**	**1**	**8**	**2**	7

Table 1.3: Nim-sequence for Kayles; Kayles(n), where $n = s + t$.

Our games will develop from the strategies analyzed in Nim and are influenced by a game called Kayles.

Game 4. *The game* Kayles *is played on a single pile of counters of size n, $n \in \mathbb{N}^{\geq 0}$, arranged in a row. On their turn, players can remove either one counter or two adjacent counters. The last player to move wins.*

Kayles was independently created by Henry Dudeney and Sam Loyd (who called the game 'Rip Van Winkle Puzzle') [4] and was later solved by Richard Guy [11]. Exemplified in Figure 1.3, we see that one slight change in the game rules of Nim can change the game outcome quite significantly. Kayles is periodic with period length 12, pre-period length 71 and the nim-sequence is depicted in Table 1.3, [4]. Table 1.3 (and all subsequence tables of this form) is read as follows: an entry corresponding to a Kayles row of length n is determined by the unique pairing of values, one from the column s and the other from row t, which sum to equal n. For example, the first entry of the period (first bolded value) corresponds to Kayles$(60 + 11)$ which is a Kayles row of 71 counters and the value of Kayles(27) = Kayles$(24 + 3)$ is 8 in the third row and the fourth column.

Variants of Kayles played on graphs have recently become popular. One variant is called Node-Kayles.

Game 5. *Node-Kayles is a game where, on their turn, players choose a vertex, which is not*

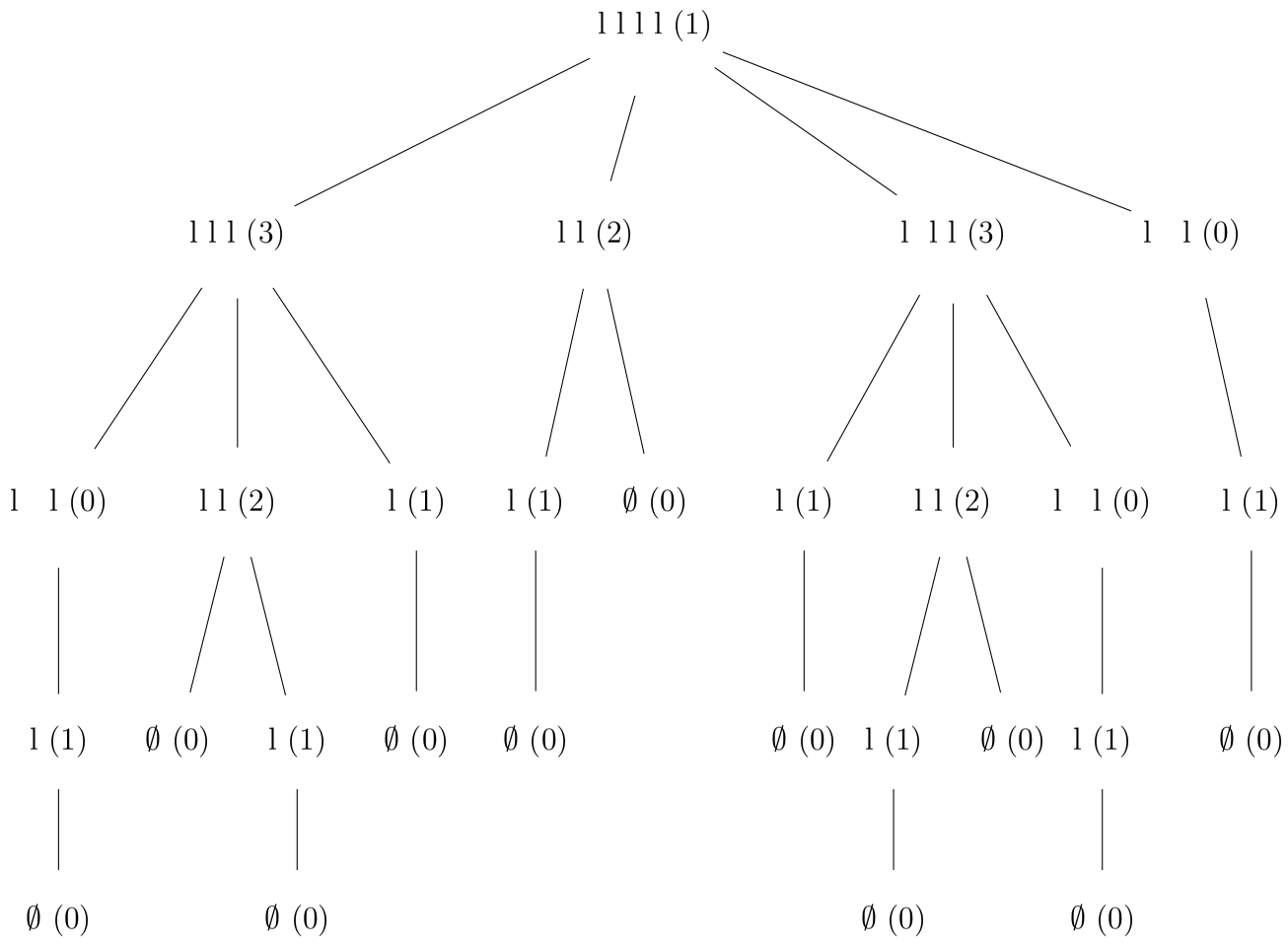

Figure 1.3: Game tree for Kayles(4).

adjacent to, and does not repeat any previously chosen vertices. This extends the independent set of chosen vertices. The last player to move completes a maximal independent set and wins the game.

This has been studied extensively in [5, 6, 16, 20] and complexity has been analyzed in [6, 28] and was proven to be P-SPACE complete by Schaefer [28]. Sprague-Grundy theory was used in [20] to analyze twelve versions of compound play of Node-Kayles on paths (six of which were under normal play). The standard version of Node-Kayles on paths has a periodic nim-sequence with period length 34, pre-period length 52. The nim-sequence of Node-Kayles on paths is shown in Table 1.4 which was originally referred to as a game called Dawson's Chess [4].

	t→	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
s ↓																		
0		0	1	1	2	0	3	1	1	0	3	3	2	2	4	0	5	2
17		2	3	3	0	1	1	3	0	2	1	1	0	4	5	2	7	4
34		0	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5
51		2	**3**	**3**	**0**	**1**	**1**	**3**	**0**	**2**	**1**	**1**	**0**	**4**	**5**	**3**	**7**	**4**
68		**8**	**1**	**1**	**2**	**0**	**3**	**1**	**1**	**0**	**3**	**3**	**2**	**2**	**4**	**4**	**5**	**5**
85		**9**	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7	4

Table 1.4: Nim-sequence for Node-Kayles on Paths [4]; Path(n), where $n = s + t$.

Much more research has been done on Node-Kayles with many different variants considered [6, 16, 20, 28]. Fleischer and Trippen [16] had studied Node-Kayles played on an interesting graph class: star graphs. Primarily motivated by [20] and [16], we study a different variant of Kayles on graph classes. This variant falls into a new class of games which we are defining to be *intersection restriction games*.

Chapter 2

Intersection Restriction Games

Throughout this paper we are interested in examining games which disallow moves based on a particular overlap of available options. We describe these games as *intersection restriction games* with the following formal definition.

Definition 5. Consider a set $X = \{1, 2, \ldots n\}$, $n \geq 1$. Let S be a set of subsets from X of size k, $k \geq 1$, and let i be fixed, where $k \geq i \geq 1$. An *intersection restriction game* is a game in which players choose elements from S such that no two chosen subsets intersect in i elements.

Example 5. *For $k = 2$, $i = 1$ we have a graph, elements of S are edges, and the intersection is a vertex. The last player to move completes a maximal matching.*

Example 6. *For $k = 3$, $i = 2$ we have a hypergraph, elements of S are hyperedges on three vertices and the intersection restriction is sharing a pair of vertices. The last player to move completes a maximal combinatorial packing design.*

We begin by studying a game which satisfies the parameters from Example 5, Arc-Kayles, on various classes of graphs. Then we look at a more complicated design theoretic packing game which satisfies the parameters from Example 6.

2.1 Arc-Kayles

Game 6. *Arc-Kayles is a combinatorial game played on a graph. On a player's turn, a player must choose an edge, remove it and its incident vertices. This in turn deletes all adjacent edges to the chosen edge. The last player to move wins.*

By the definition of Arc-Kayles, the game will terminate when we have attained a maximal matching. If the number of edges in the matching are always even or always odd we say that the winner is determined by the *parity* of the matching. The simplest solution to a game with an end condition of a maximal matching is when the underlying graph admits a parity argument for determining the winner. This type of game was highlighted in [21]. Similarly here, we have a class of graphs which already have a pre-determined solution and no combinatorial game theory strategy is necessary to determine the game play outcome. These graphs are called equimatchable graphs.

2.1.1 Equimatchable graphs

An *equiparity graph* is a graph G where all maximal matchings have the same parity. Hence, in such graphs, only the parity of the maximal matching is important to the game. If the parity of the maximal matchings of G are even, the second player has a winning move. In fact, they cannot make a mistake, so the second player will win. Otherwise, the first player will win. An equiparity graph actually has a much stronger property which will help us determine many families which have no strategy for Arc-Kayles on these graphs. We will show that if all maximal matchings in a graph have the same parity, they also have the same size. We call such graphs *equimatchable* [23]. A *path* is an acyclic graph whose vertices can be arranged in a linear sequence, where two vertices are adjacent if they are consecutive in the sequence and are non-adjacent otherwise [7]. An *M-augmenting path* is a path whose edges alternate between matching edges and non-matching edges and neither terminal edge

of the path lie in the matching [7]. We can then increase the matching size by one edge by considering this path and switching the roles between matching and non-matching edges.

Example 7. *Consider the graph G from Figure 2.1. The matching $M = \{e_2, e_4, e_8\}$ is maximal because all edges in $E(G) \setminus M$ are adjacent to edges in M. Let P be the path: $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$. P is an M-augmenting path. The edges e_1 and e_9 are the terminal edges of the path, neither lie in M and the edges of P alternate between edges from M and edges in $E(G) \setminus M$. Considering $M' = \{e_1, e_3, e_6, e_9\}$, $|M'| = |M| + 1$, and so M' is a maximal matching of larger size than M.*

Example 7 demonstrates the importance of M-augmenting paths. If we require a matching of larger size, searching for an M-augmenting path can help. If such a path is found, we can switch the edges from matching to non-matching edges (and vice versa) to get a matching of larger size.

The *symmetric difference* of two matchings from a graph G is the set of edges which do not appear in the intersection of the matchings. For example, referring to the graph G from Figure 2.1, consider two matchings: $M_1 = \{e_2, e_4, e_8\}$ and $M_2 = \{e_1, e_3, e_8\}$. The symmetric difference of M_1 and M_2 is $M' = \{e_1, e_2, e_3, e_4\}$.

Theorem 3. *Let G be a graph such that all maximal matchings have the same parity. Then G is equimatchable.*

Proof. Let G be a graph and G has the property that all maximal matchings have the same parity. Suppose M_1 and M_2 are maximal matchings and without loss of generality $|M_1| < |M_2|$. Consider the symmetric difference of M_1 and M_2. There are a few options for the components of the symmetric difference. In particular, there is a path starting and ending with edges from M_2 and alternating between M_1 and M_2. This is an M_1-augmenting path. We can switch all the edges from M_1 to M_2 and vice versa in this path and these edges (from M_2), with the rest of the edges from the matching M_1, form a matching of

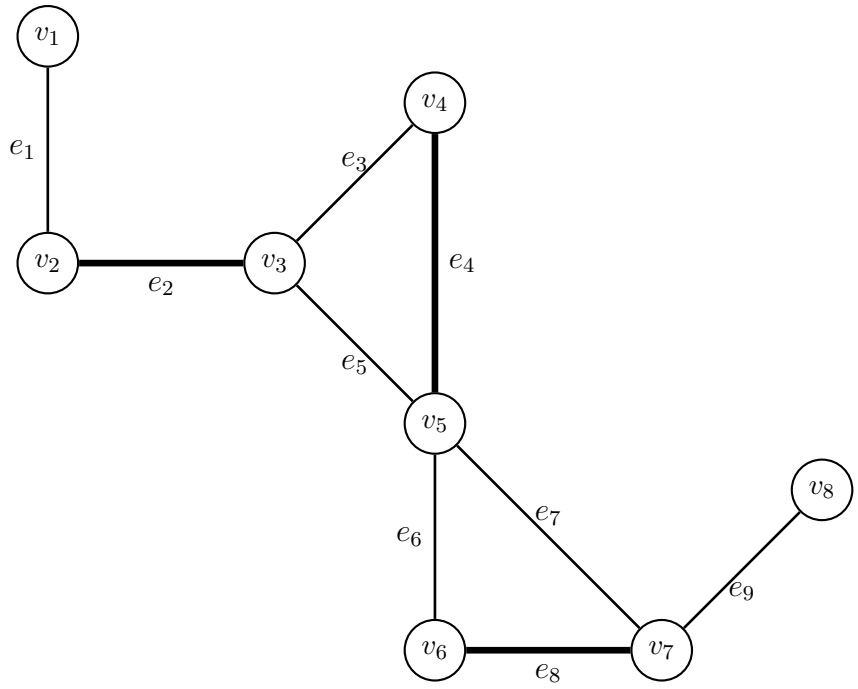

Figure 2.1: The graph G with an M-augmenting path.

opposite parity (since the augmenting path only adds one additional edge). This matching is maximal: recall that M_1 was originally maximal and we consider the augmenting path. The number of vertices which have endpoints on matching edges from M_1 increases by 2 when we switch the edges between matching and non-matching edges. We still have all the original vertices covered by the new matching edges but we now also include two extra vertices from the endpoints of the augmenting path. We increase the coverage by 2 and increase the edge count in the matching by 1. Hence, the new M_1' is still maximal and is of opposite parity which contradicts our hypothesis that G only has maximal matchings of a single parity. Therefore, if all maximal matchings have the same parity, they must also have the same size and hence are maximum and the graph G is called equimatchable.

□

Note also that since all maximal matchings are of the same size, and (in general terms)

the largest maximal matching is maximum, this means that all maximal matchings of an equimatchable graph are in fact maximum. Determining whether a graph G is equimatchable can be decided in polynomial time [12], $O(n^2m)$, in the number of vertices, n, and edges, m, of G. If G is determined to be equimatchable, then we know that all we need to do is find one maximum matching because all maximal matchings are maximum, and check its parity to determine the winner of Arc-Kayles. We can use a greedy algorithm to find a maximum matching.

This allows us to use algorithms about maximum matchings to solve some of our problems. A matching algorithm, presented in [7], relies on M-augmenting paths to find a maximum matching. Berge [2] proved a specific characterization for whether a matching is indeed maximum.

Theorem 4. [2] *A matching M in a graph G is a maximum matching if and only if G contains no M-augmenting path.*

Edmonds [14] developed a polynomial time algorithm which searches for M-augmenting paths in arbitrary graphs until the search is exhausted and then, by Berge's theorem, the resulting matching is maximum. For a succinct explanation of Edmonds' algorithm see [7], for detailed examination in narrative form, see [14]. In short, Edmonds extends an algorithm which searched for M-augmenting paths in bipartite graphs by Egerváry [7] and he deals with situations which have odd cycles and thus can be extended to arbitrary graphs.

Lesk, Plummer and Pulleyblank [23] were among the first to provide a clear classification of the graph structures for equimatchable graphs. A *perfect matching* in a graph is a matching which covers all vertices in G [7]. A graph is *randomly matchable* if every matching is a subset of a perfect matching [23]. The latter tells us that if a graph is randomly matchable, then it is equimatchable. A graph is *complete* if any two vertices are adjacent (no loops or multi-edges) [7]. The complete graph on n vertices this is denoted by K_n. A bipartite graph is a graph $G[X, Y]$ in which its vertex set can be partitioned into two subsets X and Y so that

every edge has one end in X and the other in Y [7]. A bipartite graph is complete if each vertex from X is joined to every vertex in Y. A complete bipartite graph in which $|X| = n$ and $|Y| = m$ is denoted by $K_{n,m}$.

Theorem 5. [30] *A connected graph is randomly matchable if and only if* $G = K_{n,n}$ *or* $G = K_{2n}$.

The order of a graph can forbid a perfect matching. For example, if G has an odd number of vertices it is impossible to have a perfect matching since every edge of a matching covers exactly two vertices. There can however be other properties satisfied for such cases and then G will still be equimatchable. A *factor-critical* graph is a graph where, if any one vertex is deleted, the graph has a perfect matching [7]. Let $G = (V, E)$, $S \subset V$. Let $\Gamma(S)$ be the set of vertices adjacent to vertices in S.

Theorem 6. [23] *Let* G *be a connected factor-critical graph. Then the following are equivalent:*

1. G *is equimatchable.*
2. *For every pair* u, v *of independent points and for every matching* M *of* $G \backslash \{u, v\}$, *there is a third point* w *adjacent to* u *or* v *which is not covered by* M.
3. *For every pair* u, v *of independent points in* G *there exists disjoint sets* $S_1, S_2, \ldots, S_k \subseteq \Gamma(u) \cup \Gamma(v)$ *such that*
   (a) $|S_i|$ *is odd for* $1 \leq i \leq k$;
   (b) *No line joins* S_i *to* S_j *for* $i \neq j$; *and*
   (c) $|N(S_1, S_2, \ldots, S_k)| < k$, *where*

$$N(S_1, S_2, \ldots, S_k) = \Gamma\left(\bigcup_{i=1}^{k} S_i\right) - \bigcup_{i=1}^{k} S_i - \{u, v\}.$$

Example 8. *Consider the graphs* $G = K_1 + \bigcup_{i=1}^{n} K_{2m_i}$*, where* m_i *are positive integers and* $n \geq 2$*. These are called* complete windmills *and consist of a set of* i *complete graphs on an even number of vertices* $(2m_i)$*, all of which have every vertex joined to a single center vertex, call it* d*. Take any two vertices* u *and* v *of* G *which are independent. The vertices* u *and* v *must be from two different* K_{2m_i} *(otherwise they would be adjacent). Let the set of neighbours of* u *without* d *be* S_u*, and similarly let the set of neighbours of* v *without* d *be* S_v*. Then* $|S_u|$ *is odd,* $|S_v|$ *is odd and* $S_u \cap S_v = \emptyset$*. Since* d *is a cut-vertex of* G *located between* S_u *and* S_v *there is no edge between the two sets of vertices* S_u *and* S_v *for all* $u \neq v$*. And since* $N(S_u, S_v) = \{d\}$*,* $|N(S_u, S_v)| = 1 < 2$ *and* G *satisfies all the conditions of Theorem 6 part* 3*, and hence is equimatchable. Note that if* $n = 1$ *we have a complete graph* $G = K_{2m_i+1}$*. This too is factor-critical, and equimatchable with* m_i *edges in a maximum matching.*

Theorem 7. [23] *A connected bipartite graph* $G = G[U, W]$ *with* $|U| \leq |W|$*, is equimatchable if and only if, for all* $u \in U$*, there exists a non-empty* $X \subseteq \Gamma(u)$ *such that* $|\Gamma(X)| \leq |X|$*.*

These authors also present other characterizations for connected graphs without a perfect matching and are not factor-critical. We omit these classifications here as they rely on very technical proofs and stating their theorems would require extensive introduction to new terminology. We encourage the interested reader to see [23] for these details.

Example 9. C_5*, the cycle graph on five vertices, is an equimatchable graph.* C_5 *has five vertices and five edges. Playing Arc-Kayles on* C_5*, any starting position is isomorphic and eliminates three edges. This leaves a path on three vertices. Either choice of edge will eliminate the remainder of the edges. Hence the size of any maximal matching of* C_5 *is two and the graph is equimatchable.*

Proposition 4. [17] *If* $G \cong C_k$*, then* G *is equimatchable if and only if* $k \in \{3, 4, 5, 7\}$*.*

We in fact have much more than this classification. The *girth* of a graph G is the size of the smallest cycle in G. A *leaf* is a vertex of degree 1. A *stem* is a vertex which is

connected to a leaf [7]. Frendrup, Hartnell and Vestergaard [17] also showed that if a graph is equimatchable with girth at least five, then the graph is of a particular form. They define a special family of graphs as follows: let $\mathcal{F}$ represent the set of graphs including K_2 and connected bipartite graphs with vertex sets V_1 and V_2 where all vertices in V_1 are stems and V_2 are all vertices which are not stems.

Lemma 3. [17] *Each graph from $\mathcal{F}$ is equimatchable.*

Theorem 8. [17] *Let G be a connected equimatchable graph with girth $g(G) \geq 5$. Then $G \in \mathcal{F} \cup \{C_5, C_7\}$.*

Graph connectivity plays a role in two works on equimatchable graphs. A graph is *k-connected* if for any two vertices u and v in a graph G there are k internally disjoint paths from u to v [7]. A *cut-vertex* is a vertex which, if deleted, will increase the number of connected components [7].

Theorem 9. [15] *A 1-connected graph is an equimatchable factor-critical graph if and only if:*

1. *There exists exactly one cut-vertex d.*
2. *Every connected component C_i of $X \setminus \{d\}$ is isomorphic to K_{2r_i} or to K_{r_i,r_i}.*
3. *d is adjacent to at least two adjacent vertices of each C_i.*

Lemma 4. [15] *Let G be an equimatchable factor-critical graph of order at least 4 and connectivity 2, and let $S = \{s_1, s_2\}$ be a 2-vertex cut of G. Then $X \setminus S$ has precisely two components one of which is odd and one of which is even.*

Favaron then proceeds to give a rather technical theorem, which gives the structure of graphs satisfying the properties of Lemma 4.

Case analysis on a few graph classes was undertaken by Kawarabayashi, Plummer and Saito [22] in 2003. A *planar graph* is a graph G which can be drawn in a plane where edges of G only meet at end vertices [7]. Kawarabayashi, Plummer and Saito [22] prove that if a graph G is 3-connected, planar and equimatchable, then $|V(G)| \leq 9$ and must be one of the 23 graphs they list.

A *cubic graph* is a graph in which every vertex has degree three [7]. Using the Gallai-Edmonds Theorem [24], Kawarabayashi et al. prove the following:

Theorem 10. [22] *If G is a connected cubic equimatchable graph then G is either isomorphic to K_4 or $K_{3,3}$.*

We now turn our attention to graphs which are not equimatchable.

2.1.2 Paths

A *path* is a connected, acyclic graph in which all vertices have degree at most two. Arc-Kayles on paths is the simplest non-trivial version of the game on a graph. Players play alternately on a path of length n, where n is the number of vertices. This is a non-standard definition for the length of a path; it is used here to simplify our discussion. Removing end edges results in a path of length $n-2$; removing intermediate edges results in a disjunctive sum of paths. The last player to move wins.

We have programmed this game into the SAGE open source mathematical software through a server at Carleton University. The server hardware includes 2 Intel Xeon E5667 processors with a total of 12 processor cores at a clockrate speed of 3.30 GHz. There is 128GB of RAM which runs at a speed of 1800MHz.

The nim-sequence obtained for Arc-Kayles on paths is shown in Table 2.1. The first period is in bold font. The pre-period length is 53 and the period length is 34.

This game has the same period as that of Berlekamp et al. [4] for Node-Kayles on paths.

	t→	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
s ↓																		
0		0	0	1	1	2	0	3	1	1	0	3	3	2	2	4	0	5
17		2	2	3	3	0	1	1	3	0	2	1	1	0	4	5	2	7
34		4	0	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5
51		5	2	**3**	**3**	**0**	**1**	**1**	**3**	**0**	**2**	**1**	**1**	**0**	**4**	**5**	**3**	**7**
68		**4**	**8**	**1**	**1**	**2**	**0**	**3**	**1**	**1**	**0**	**3**	**3**	**2**	**2**	**4**	**4**	**5**
85		**5**	**9**	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7

Table 2.1: Nim-sequence for Arc-Kayles on Paths; Path(n), where $n = s + t$.

This makes sense intuitively because in Node-Kayles we are deleting the vertex and its neighbours, in Arc-Kayles we are deleting an edge and its terminal vertices (in turn, deleting adjacent edges). This has the same effect as taking a path of length $n + 1$ for Arc-Kayles and a path of length n for Node-Kayles and superimposing the vertices of the path of length n on the edges of the path of length $n + 1$ in a one-to-one correspondence. This generalizes to the next theorem. A *line graph* of a graph G is the graph produced from switching all edges of G to vertices and these vertices are connected by an edge if the corresponding edges were adjacent in G [7]. Choosing an edge in G corresponds exactly to choosing a vertex in the line graph of G. The proof of the theorem is immediate by the above discussion.

Theorem 11. *Arc-Kayles on G is Node-Kayles on the line graph of G.*

We extend the study of paths to graphs which have paths as sub-positions, first starting with star graphs, then cycle graphs and wheel graphs.

2.1.3 Stars

A *star* graph is a connected, acyclic graph with one distinguished node, emanating from which are a finite number of rays. The distinguished node can have degree n, $n \geq 1$, while all other nodes have degree at most two. If the distinguished node has degree one or two, the star is a path. This is a non-standard definition for stars; here we follow the notation

from [16].

Motivated by a paper by Fleischer and Trippen [16], we first provide a decomposition for Arc-Kayles on a general star of three rays, followed by results.

Let S_{l_1,l_2,l_3} represent a star graph with rays of length l_1, l_2 and l_3. In building a star from the three paths, the three paths (rays) are joined at one terminal vertex of each of the rays. For example, $S_{l_1,1,1}$ is a path on l_1 vertices, while $S_{l_1,2,2}$ has a distinguished center vertex of degree 3, with two rays of length two, each have one edge, and the third ray has length l_1 with $l_1 - 1$ edges. A path of length zero has no real meaning and hence we look at stars with rays of non-trivial length (i.e., length one or greater). Recall we denote a path with n vertices as $Path\,(n)$.

Let us begin by providing the general decomposition for Arc-Kayles on stars. From a general star S_{l_1,l_2,l_3} we have the following options.

1. A player may remove an edge incident to the center node. This can happen in three ways and produces the following three outcomes:

   (a) $Path\,(l_1 - 2) \oplus Path\,(l_2 - 1) \oplus Path\,(l_3 - 1)$

   (b) $Path\,(l_1 - 1) \oplus Path\,(l_2 - 2) \oplus Path\,(l_3 - 1)$

   (c) $Path\,(l_1 - 1) \oplus Path\,(l_2 - 1) \oplus Path\,(l_3 - 2)$

2. A player may remove the second edge from the center vertex on any of the rays. These will give the following three outcomes:

   (a) $Path\,(l_1 - 3) \oplus Path\,(l_2 + l_3 - 1)$

   (b) $Path\,(l_2 - 3) \oplus Path\,(l_1 + l_3 - 1)$

   (c) $Path\,(l_3 - 3) \oplus Path\,(l_1 + l_2 - 1)$

3. A player may remove any other edge from any the rays. These will leave a disjunctive sum of a star and a path:

(a) $S_{l_1-i-2,l_2,l_3} \oplus Path\,(i)\,, 0 \leq i \leq l_1 - 2$

(b) $S_{l_1,l_2-i-2,l_3} \oplus Path\,(i)\,, 0 \leq i \leq l_2 - 2$

(c) $S_{l_1,l_2,l_3-i-2} \oplus Path\,(i)\,, 0 \leq i \leq l_3 - 2$

Each one of these sets S_i of moves contributes to the mex which computes the value of position S_{l_1,l_2,l_3}.

The first non-trivial class of stars to examine are stars defined as follows: let $l_1 = 2$, fix $l_2 \geq 2$ and lastly let $l_3 = n$ vary, going to infinity. Results from [16] for Node-Kayles showed that the pre-periods for this class of graphs for several values of the second ray modulo 34 are not bounded, and hence there is no known closed form for Node-Kayles on star graphs. Table 2.2 shows that for Arc-Kayles, the pre-periods appear to be eventually bounded. Let us take a closer look as this star class. $S_{2,2,l_3}$ is equivalent to a path on $l_3 + 1$ vertices. This is for these reasons: first, introducing an additional ray of length 2 does not add any new moves. Removing it, or the other fixed ray of length 2 deletes both rays and one edge off of the ray of length l_3. Second, this is true because $\mathscr{G}\,(Path\,(2)) = \mathscr{G}\,(Path\,(3)) = 1$ and so when we delete the second last edge from the ray of length l_3 (closest to the distinguished center node), no new moves are added from $Path\,(2)$ to $Path\,(3)$ and hence the values are the same, admitting the same result if it were a path or a star under these ray length restrictions.

Of course, as the length of l_2 increases, the analysis is not as straightforward. We summarize this as a general star in the following theorem, followed by this special case as a corollary. As seen in the general star decomposition there are many parts contributing to the periodicity. For notational simplicity, we consider $i_{p_{i,j,n}}$ to be the pre-period of the path associated with the star $S_{i,j,n}$, as n varies, going to infinity.

Theorem 12. *Let l_1 and l_2 be positive integers. Suppose $S_{l_1',l_2',i'}$ is periodic, where $l_1' \leq l_1$ and $l_2' \leq l_2$, and $S_{l_1,l_2,i'}$ is computed up to value $i' = n$ such that we observe a pre-period of*

size i_s and periodicity thereafter with period length p'. If $n > i + 2p + 2$ where

$$i = max\left\{i_s + i_{p_{l_1,l_2,n}}, i_{p_{l_1,l_2-j-2,n}} - p, i_{path}\right\}$$

and

$$p = LCM\left\{p', Period\left\{S_{l_1,l_2-2-j,n}\right\}_{j=0}^{l_2-2}, Period\left\{S_{l_1-2-j,l_2,n}\right\}_{j=0}^{l_1-2}, p_{path}\right\}$$

then $S_{l_1,l_2,i}$ is periodic with pre-period i_s and period p'.

Proof. Recall that the path sequence is periodic (presented in subsection 2.1.2). The value of a star $S_{l_1,l_2,n}$ with ray lengths l_1 and l_2 fixed and n varying to infinity, is determined by the minimum excluded value of its sub-positions. We want to show that if n is large enough and $\mathscr{G}\left(S_{l_1,l_2,n}\right) = mex\left(T\right)$ and $\mathscr{G}\left(S_{l_1,l_2,n-p}\right) = mex\left(S\right)$, then $T = S$. Hence the following sequences need to be periodic and shifting n back by p we have the same nim-value for each sequence:

1. $Path\left(l_1 - 2\right) \oplus Path\left(l_2 - 1\right) \oplus Path\left(n - 1\right)$
2. $Path\left(l_1 - 1\right) \oplus Path\left(l_2 - 2\right) \oplus Path\left(n - 1\right)$
3. $Path\left(l_1 - 1\right) \oplus Path\left(l_2 - 1\right) \oplus Path\left(n - 2\right)$
4. $Path\left(l_1 - 3\right) \oplus Path\left(l_2 + n - 1\right)$
5. $Path\left(l_2 - 3\right) \oplus Path\left(l_1 + n - 1\right)$
6. $Path\left(n - 3\right) \oplus Path\left(l_1 + l_2 - 1\right)$
7. $S_{l_1-i-2,l_2,n} \oplus Path\left(i\right), 0 \leq i \leq l_1 - 2$
8. $S_{l_1,l_2-i-2,n} \oplus Path\left(i\right), 0 \leq i \leq l_2 - 2$
9. $S_{l_1,l_2,n-i-2} \oplus Path\left(i\right), 0 \leq i \leq n - 2$

We now check that each sub-position is periodic. First consider sub-positions $1, 2, 3$ as listed above. Note: i_{path} is the pre-period of the path sequence and p_{path} is the period of the path sequence. These sub-positions are paths where two of the summands of paths are fixed and the third (involving n) is varying. This is equivalent to taking the path of the nim-sequence (which we know to be periodic) and adding the same constant to every entry. If n is larger than $i_{path} + p_{path} + 1$, then $\mathscr{G}(Path(n-1)) = \mathscr{G}(Path(n-p-1))$, and so $\mathscr{G}(Path(n-1)) \oplus \mathscr{G}(Path(l_1-2)) \oplus \mathscr{G}(Path(l_2-1)) = \mathscr{G}(Path(n-p-1)) \oplus \mathscr{G}(Path(l_1-2)) \oplus \mathscr{G}(Path(l_2-1))$. These nim-sequences are periodic with period length 34. The same holds for sub-position 2 (indices are switched and the proof is the same). For sub-position 3, n must be larger than the $i_{path} + p_{path} + 2$. This shows that when n is sufficiently large, the Grundy values of the first three positions are all contained in both T and in S.

For sub-positions $4, 5$ involving $n + l_i - 1, i = 1, 2$ this is a path sequence shifted by $l_i - 1$. The second component of the nim-addition is a fixed length path and so the same explanation as the first set $\{1, 2, 3\}$ of path sub-positions applies. Lastly, the sixth sub-position consists of a path sequence (shifted by three) being nim-added to a constant. This also reduces to the earlier cases with the slight modification that n must be as large as $i_{path} + p_{path} + 3$. All three sub-positions are periodic with period length 34. Once again, n must be larger than $i_{path} + p_{path} + 3$. And so, the Grundy values of the three sub-positions $\{4, 5, 6\}$ are in both T and S.

The seventh and eighth sub-positions listed rely on the breakdown of the fixed ray of lengths l_1 and l_2 respectively. We examine the seventh subposition in detail, the eighth is symmetric. There are finitely many sub-positions of the form $\{S_{l_1-2-j,l_2,n} \oplus Path(j)\}_{j=0}^{l_1-2}$. The stars considered in this set rely on different star classes with respect to l_2 which we consider to have already been determined. Hence, in order to determine the periodicity of the current class, we need to be large enough with respect to all of the previously calculated

classes already attaining periodicity and the periodicity of all classes involved have to be synchronized. And so, we need

$$n \geq max\left\{preperiod\left\{S_{l_1-j-2,l_2,n}\right\}_{j=0}^{l_1-2}\right\} + LCM\left\{period\left\{S_{l_1-j-2,l_2,n}\right\}_{j=0}^{l_1-2}\right\}$$

and

$$n \geq max\left\{preperiod\left\{S_{l_1,l_2-j-2,n}\right\}_{j=0}^{l_2-2}\right\} + LCM\left\{period\left\{S_{l_1,l_2-j-2,n}\right\}_{j=0}^{l_2-2}\right\}.$$

If n is sufficiently large then the Grundy values will be contained in both T and S.

Lastly, the first terms of the sub-positions $S_{l_1,l_2,n-2-j} \oplus Path(j), 0 \leq j \leq n-2$ remain within the same star class and we look back at previously calculated positions and nim-add a path of the appropriate length. We consider two cases:

Case 1: Suppose $n-j-2 > i_s+p$. Since $n-j-2 > i_s+p$, $n-j-2-p > i_s$, the sequence of $S_{l_1,l_2,n}$ is already showing signs of periodicity. And so, $\mathscr{G}\left(S_{l_1,l_2,l_{n-2-j-p}}\right) = \mathscr{G}\left(S_{l_1,l_2,l_{n-2-j}}\right)$. This implies that adding a constant to both positions will give the same value. Hence $\mathscr{G}\left(S_{l_1,l_2,l_{n-2-j-p}} \oplus Path\left(j\right)\right) = \mathscr{G}\left(S_{l_1,l_2,l_{n-2-j}} \oplus Path\left(j\right)\right)$. Thus the Grundy values of these positions appear in both T and S.

Case 2: Suppose $n-j-2 \leq i_s+p$. If $n > i+2p+2$, then $j > i_{path}+p$ and $\mathscr{G}\left(Path\left(j-p\right)\right) = \mathscr{G}\left(Path\left(j\right)\right)$. Then $\mathscr{G}\left(S_{l_1,l_2,n-2-j} \oplus Path\left(j-p\right)\right) = \mathscr{G}\left(S_{l_1,l_2,n-2-j} \oplus Path\left(j\right)\right)$. Thus the Grundy values of these positions appear in both T and S.

When $n > i+2p+2$, we conclude that $S = T$, where

$$i = max\left\{i_s + i_{p_{l_1,l_2,n}}, i_{p_{l_1,l_2-j-2,n}} - p, i_{path}\right\}$$

and

$$p = LCM\left\{p', Period\left\{S_{l_1,l_2-2-j,n}\right\}_{j=0}^{l_2-2}, Period\left\{S_{l_1-2-j,l_2,n}\right\}_{j=0}^{l_1-2}, p_{path}\right\}.$$

□

Corollary 2. *Let ray lengths $l_1 = 2$, l_2 be fixed and $l_3 = n$ as n goes to infinity. If $S_{2,l_2,n}$ is believed to be periodic, we need to compute only up to a certain value of $l_3 = n, n \in \mathbb{N}, n > i + 2p + 2$, where*

$$i = max\left\{i_s + i_{p_{2,l_2,n}}, i_{p_{2,l_2-j-2,n}} - p, i_{path}\right\}$$

and

$$p = LCM\left\{Period\left\{S_{2,l_2,n-2-j}\right\}_{j=0}^{n-2}, Period\left\{S_{2,l_2-2-j,n}\right\}_{j=0}^{l_2-2}, p_{path}\right\}.$$

The proof of Corollary 2 is immediate from Theorem 12. We include Corollary 2 because we have examined these families of graphs. By Table 2.2 it seems as though the pre-periods for all star classes modulo 34 with a fixed ray $l_1 = 2$ eventually stabilize.

This preliminary data leads us to the following conjecture.

The *quasi-preperiod* for a star with a ray of length $l_2 \pmod{34}$ is the value of l_2 at which the pre-period for that class of stars becomes stable in l_3.

Conjecture 1. *For all $l_2 >$ quasi-preperiod, and $l_3 >$ pre-period shown in Table 2.2, the Grundy value of Arc-Kayles played on S_{2,l_2,l_3} is equal to value shown in Table 2.3.*

If Conjecture 1 is true, it means that there is indeed an ultimate star $S_{2,l_2,n}$ which is completely determined by smaller star classes (as was hoped for Node-Kayles in [16]) for Arc-Kayles. At this time, it is unclear how to prove this conjecture as even the first step of proving boundedness a priori of the nim-sequence is hard. This will be further discussed in the future directions section. We now turn our attention to cycle graphs.

2.1.4 Cycles

Cycles are closed paths. Their characterization is relatively simple now that we know that paths are periodic. As is commonplace in graph theory literature, we denote a cycle of length

	t→	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
s↓																																			
0		52	52	52	196	52	156	349	232	301	191	249	181	190	383	232	188	388	190	283	170	250	261	349	125	349	249	162	191	249	177	303	178	245	232
34		349	249	249	249	215	245	216	250	165	256	250	388	258	248	249	250	261	197	249	245	388	246	249	383	349	249	241	204	284	212	245	250	349	245
68		279	249	283	204	241	260	383	301	165	261	459	388	301	232	283	279	244	349	388	258	260	283	245	383	349	388	260	301	349	246	250	333	314	264
102		336	320	283	204	283	260	349	248	285	299	388	388	294	232	249	250	272	316	283	258	245	280	280	250	396	388	294	459	349	333	250	314	349	320
136		313	**388**	312	228	283	396	349	313	279	283	388	388	260	232	343	333	271	306	303	272	250	280	301	250	396	388	273	459	303	459	298	300	349	388
170		313	**388**	459	221	283	396	349	345	313	283	388	388	396	232	299	252	373	459	283	459	333	333	396	**388**	396	286	340	461	383	459	250	**333**	358	459
204		316	**388**	459	**340**	261	275	349	388	313	283	275	388	**459**	232	303	461	280	374	283	258	301	280	260	**388**	349	255	336	461	383	459	298	**333**	**459**	367
238		333	**388**	326	**340**	261	**338**	349	**459**	301	283	**388**	320	**459**	232	367	461	340	326	341	258	301	283	459	**388**	**459**	337	329	**388**	383	301	298	**333**	**459**	**459**
272		299	**388**	**459**	**340**	**283**	**338**	349	**459**	313	**351**	**388**	**349**	**459**	374	316	337	408	348	341	256	**392**	**261**	**443**	**388**	**459**	303	341	**388**	383	301	298	**333**	**459**	**459**
306		299	**388**	**459**	**340**	**283**	**338**	349	**459**	313	**351**	**388**	**349**	**459**	340	303	**461**	**340**	**313**	341	256	**392**	**261**	**443**	**388**	**459**	**255**	374	**388**	**303**	**392**	**284**	**333**	**459**	**459**
340		299	**388**	**459**	**340**	**283**	**338**	**392**	**459**	313	**351**	**388**	**349**	**459**	374	303	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	341	**388**	**303**	**392**	**284**	**333**	**459**	**459**
374		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	374	303	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
408		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
442		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
476		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
510		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
544		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
578		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**
612		**461**	**388**	**459**	**340**	**283**	**338**	**392**	**459**	**301**	**351**	**388**	**349**	**459**	**340**	**329**	**461**	**340**	**313**	**443**	**309**	**392**	**261**	**443**	**388**	**459**	**255**	**374**	**388**	**303**	**392**	**284**	**333**	**459**	**459**

Table 2.2: Preperiods for Star graphs with 3 rays; $l_1 = 2$, $l_2 = s + t$ and $l_3 = n$ as n goes to infinity.

	$l_3\rightarrow$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$l_2\downarrow$																																			
0		6	8	8	1	26	4	28	3	13	29	41	42	27	13	33	12	8	1	45	29	3	24	50	26	48	15	54	14	10	12	4	38	5	30
1		8	1	1	9	0	23	8	26	0	4	23	2	6	22	15	5	5	9	3	3	34	5	21	3	16	2	6	18	4	0	8	5	21	19
2		8	1	1	2	0	37	1	43	0	28	52	20	18	4	38	5	43	39	3	50	27	1	23	41	36	13	51	47	4	4	5	40	7	4
3		1	9	2	3	14	26	2	4	8	14	5	20	4	8	5	18	41	3	8	5	13	3	28	39	5	14	4	7	5	6	14	15	39	18
4		26	0	0	14	1	2	4	14	14	5	43	4	16	5	31	4	4	24	2	19	36	0	33	2	38	4	24	6	5	9	4	2	6	5
5		4	23	37	26	2	4	12	6	13	7	4	42	5	9	33	7	19	2	9	4	16	28	13	13	4	8	20	5	10	7	8	42	5	40
6		28	8	1	2	4	12	1	29	7	4	8	2	17	4	4	21	5	51	3	15	37	5	6	4	31	19	1	9	0	8	8	5	40	15
7		3	26	43	4	14	6	29	4	12	46	14	44	7	58	49	13	43	43	54	44	37	29	18	4	5	9	52	52	17	46	1	48	48	20
8		13	0	0	8	14	13	7	12	8	5	26	23	3	5	5	4	4	12	2	6	9	4	8	31	1	10	7	8	5	5	9	11	22	13
9		29	4	28	14	5	7	4	46	5	5	53	7	24	1	48	4	0	31	1	17	26	7	36	14	18	32	20	6	5	38	7	35	13	6
10		41	23	52	5	43	4	8	14	26	53	3	11	5	34	15	12	2	13	22	4	7	27	6	13	4	20	25	47	35	12	7	19	14	0
11		42	2	20	20	4	42	2	44	23	7	11	1	18	19	0	29	1	35	4	0	42	6	27	0	26	5	5	43	3	3	36	1	23	4
12		27	6	18	4	16	5	17	7	3	24	5	18	4	19	14	2	39	40	4	41	27	40	7	23	5	13	26	4	8	14	14	36	0	38
13		13	22	4	8	5	9	4	58	5	1	34	19	19	2	2	26	4	12	14	6	9	3	8	39	5	7	12	19	1	13	0	4	2	45
14		33	15	38	5	31	33	4	49	5	48	15	0	14	2	3	3	21	1	1	30	4	7	50	5	16	7	42	32	2	0	4	17	30	2
15		12	5	5	18	4	7	21	13	4	4	12	29	2	26	3	1	1	43	38	7	8	9	5	38	4	15	50	13	21	0	36	9	3	34
16		8	5	43	41	4	19	5	43	4	0	2	1	39	4	21	1	1	25	4	7	42	1	40	7	39	44	2	0	12	28	2	6	47	46
17		1	9	39	3	24	2	51	43	12	31	13	35	40	12	1	43	25	7	12	41	54	40	12	41	19	10	16	12	28	48	6	40	35	37
18		45	3	3	8	2	9	3	54	2	1	22	4	4	14	1	38	4	12	5	1	9	3	8	2	9	30	4	20	6	13	26	4	29	5
19		29	3	50	5	19	4	15	44	6	17	4	0	41	6	30	7	7	41	1	26	32	3	3	13	22	0	19	6	6	31	4	41	5	5
20		3	34	27	13	36	16	37	37	9	26	7	42	27	9	4	8	42	54	9	32	3	31	1	0	4	12	34	5	7	4	21	5	14	43
21		24	5	1	3	0	28	5	29	4	7	27	6	40	3	7	9	1	40	3	3	31	1	19	12	28	6	26	26	4	4	5	9	20	3
22		50	21	23	28	33	13	6	18	8	36	6	27	7	8	50	5	40	12	8	3	1	19	7	28	6	9	42	4	15	5	5	51	7	50
23		26	3	41	39	2	13	4	4	31	14	13	0	23	39	5	38	7	41	2	13	0	12	28	5	6	29	3	8	2	9	3	3	36	5
24		48	16	36	5	38	4	31	5	1	18	4	26	5	5	16	4	39	19	9	22	4	28	6	6	7	4	36	5	9	34	3	32	6	5
25		15	2	13	14	4	8	19	9	10	32	20	5	13	7	7	15	44	10	30	0	12	6	9	29	4	8	5	9	32	12	34	13	13	12
26		54	6	51	4	24	20	1	52	7	20	25	5	26	12	42	50	2	16	4	19	34	26	42	3	36	5	6	4	7	33	5	48	38	56
27		14	18	47	7	6	5	9	52	8	6	47	43	4	19	32	13	0	12	20	6	5	26	4	8	5	9	4	4	31	13	25	43	41	38
28		10	4	4	5	5	10	0	17	5	5	35	3	8	1	2	21	12	28	6	6	7	4	15	2	9	32	7	31	1	26	9	37	42	14
29		12	0	4	6	9	7	8	46	5	38	12	3	14	13	0	0	28	48	13	31	4	4	5	9	34	12	33	13	26	1	15	2	2	10
30		4	8	5	14	4	8	8	1	9	7	7	36	14	0	4	36	2	6	26	4	21	5	5	3	3	34	5	25	9	15	2	6	13	28
31		38	5	40	15	2	42	5	48	11	35	19	1	36	4	17	9	6	40	4	41	5	9	51	3	32	13	48	43	37	2	6	10	4	4
32		5	21	7	39	6	5	40	48	22	13	14	23	0	2	30	3	47	35	29	5	14	20	7	36	6	13	38	41	42	2	13	4	8	35
33		30	19	4	18	5	40	15	20	13	6	0	4	38	45	2	34	46	37	5	5	43	3	50	5	5	12	56	38	14	10	28	4	35	5

Table 2.3: Grundy values for stars S_{2,l_2,l_3}, with $l_1 = 2$, $l_2 >$ quasi-preperiod, $l_3 >$ preperiod shown in Table 2.2.

	t→	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
s↓																		
0		0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
17		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
34		0	0	0	1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
51		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
68		**0**	**0**	**0**	**0**	0	0	0	1	0	0	0	1	0	0	0	0	0

Table 2.4: Nim-sequence for Arc-Kayles on the cycle graph; Cycle(n), where $n = s + t$.

n as $\mathcal{C}_n$, where n is the number of vertices in the cycle. The graph is *edge transitive*, meaning that choosing any edge for the next player's initial move is equivalent to choosing any other edge. This implies that the value for $\mathcal{C}_n$ is completely determined by the mex of Path($n-2$). And so, the nim-sequence for cycles is binary and is the path sequence, shifted by two, with all non-zero values becoming zero, and all zeros becoming one. For cycles, the sequence is periodic with pre-period of length 38, and period length 34. The cycle nim-sequence is shown in Table 2.4.

A natural extension of the cycle graph is the wheel graph. We explore this graph class next.

2.1.5 Wheels

A *wheel* is a connected graph with one distinguished center node, surrounded by a cycle where every node on the cycle is connected by one edge to the center node.

There are two moves for a general wheel graph, $Wheel\,(n)$, where n is the number of vertices in the cycle surrounding the distinguished node and $n \geq 3$. One move is to choose a spoke edge, which leaves a path of length $n-1$. The other move is to choose a rim edge which leaves a fan graph with $n-2$ vertices on the rim. We refer to the fan graph as a *pizza graph* since further breakdown of sub-positions by removing more rim edges from a fan resembles a partially eaten pizza. Note: the values for $\mathscr{G}\,(Wheel\,(0))\,, \mathscr{G}\,(Wheel\,(1))$ and

$\mathscr{G}(Wheel(2))$ are by convention 0 since we are not allowing for multi-edges: the graphs have no edges and the next player does not have a move.

Lemma 5. *Let $n \geq 3$. $\mathscr{G}(Pizza(n-2)) \geq \mathscr{G}(Path(n-1))$.*

Proof. We begin by breaking down the options of each game.
From $Path(n-1)$, a player may remove any edge. It may disconnect the path into two disjoint paths. The set of moves are summarized as the set of disjunctive sums of paths $\{Path(n-3-i) \oplus Path(i)\}$, for $0 \leq i \leq n-3$.

From $Pizza(n-2)$ there are two possible moves: a player may remove a spoke edge and obtain the same set of disjunctive sums of paths as from $Path(n-1)$:
$\{Path(n-3-i) \oplus Path(i)\}$, for $0 \leq i \leq n-3$. Otherwise, a player may remove a rim edge and we obtain another pizza with one or two separate smaller fans joined at the distinguished center node. This set of positions looks like the following set: $\{Pizza(n-i-4, i)\}$, for $0 \leq i \leq n-4$, where $n-i-4$ and i represent the sizes of the rim edge paths of the fans.

Since $Pizza(n-2)$ has as one of its set of sub-positions identical to all of the sub-positions of $Path(n-1)$, all these options are also contributing to the mex of $Pizza(n-2)$. Therefore, $\mathscr{G}(Pizza(n-2))$ is at least as large as $\mathscr{G}(Path(n-1))$. □

Theorem 13.

$$\mathscr{G}(Wheel(n)) = \begin{cases} 1 & \text{if } \mathscr{G}(\text{Path}(\text{n}-1)) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Proof. The game $Wheel(n)$ can have value at most 2 because in order to calculate the value of $Wheel(n)$ we use the mex function on its options. Up to isomorphism there are only two options from any complete wheel graph and hence we are only taking the mex of two values:

i	j	$\mathscr{G}(j)$	$\mathscr{G}(i)$ required for $\mathscr{G}(path(i)) \oplus \mathscr{G}(path(j)) = 1$
n-3	0	0	1
n-4	1	0	1
n-5	2	1	0
n-6	3	1	0
n-7	4	2	3
n-8	5	0	1
n-9	6	3	2
n-10	7	1	0

Table 2.5: $Wheel(n)$.

$mex(\mathscr{G}(Pizza(n-2)), \mathscr{G}(Path(n-1)))$. By Lemma 5, we know that $\mathscr{G}(Pizza(n-2)) \geq \mathscr{G}(Path(n-1))$. There are three possibilities:

1. $\mathscr{G}(Pizza(n-2)) \geq \mathscr{G}(Path(n-1)) > 0$ implying that $\mathscr{G}(Wheel(n)) = 0$.

2. $\mathscr{G}(Pizza(n-2)) > 1 > \mathscr{G}(Path(n-1)) = 0$ or $\mathscr{G}(Pizza(n-2)) = \mathscr{G}(Path(n-1)) = 0$ implying that $\mathscr{G}(Wheel(n)) = 1$.

3. $\mathscr{G}(Pizza(n-2)) = 1, \mathscr{G}(Path(n-1)) = 0$ implying that $\mathscr{G}(Wheel(n)) = 2$.

We show that Case 3 cannot occur. If $\mathscr{G}(Path(n-1)) = 0$ we know all of its sub-positions are greater than 0. Recall that the sub-positions of $Path(n-1)$ are also sub-positions of $Pizza(n-2)$. If we can show that all the paths which have value 0 will have a sub-position equal to 1 we will have completed the proof.

We can use Grundy scales to check the values. When a position involves a connected path on $n-1$ vertices, we know its sub-positions will involve a set of disjunctive sums of paths which are on $n-3$ vertices and possibly split into two pieces. Looking at Table 2.5 we determine the necessary values to obtain a 1 for $\mathscr{G}(Path(i) \oplus Path(n-i-3))$. Recall that n refers to the number of vertices on the rim of the wheel.

We align the values for paths along the first strip of paper from left to right. Then we align on the second piece of paper in reverse order with the last column of Table 2.5. We line

i							n-9	n-8	n-7	n-6	n-5	n-4	n-3
$\mathscr{G}(Path(i))$							2	1	3	0	0	1	1
$\mathscr{G}(Path(m))$	0	1	1	2	0	3	1	1	0	3	3	2	2
m	1	2	3	4	5	6	7	8	9	10	11	12	13

Table 2.6: Grundy value check for $n = 16$.

	t→	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
s↓																		
0		0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
17		0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
34		0	0	1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
51		**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
68		**0**	**0**	**0**	0	0	0	1	0	0	0	1	0	0	0	0	0	0

Table 2.7: Nim-sequence for Wheel(n), where $n = s + t$.

up the two papers as seen in Table 2.6 in the calculation for $n = 16$, hence $Path(15)$. If there is at least one value that matches, then $Path(n-1)$ has a 1 as a sub-position. In Table 2.6 we see that when $i = n-8 = 16-8 = 8$ the required path value equals the actual path value. Hence, Path(15) has a sub-position equal to 1, at sub-position $Path(8) \oplus Path(5)$. This task is repeated for all 0 valued paths in the nim-sequence for paths within the pre-period and first full preriod. Every 0 position has at least one sub-position with value 1 implying that $Pizza(n-2)$ also has at least one sub-position with value of 1. Therefore, Case 3 cannot occur and the value of $Wheel(n)$ is binary, periodic and completely determined by $Path(n-1)$. The nim-sequence for $Wheel(n)$ is summarized in Table 2.7.

□

Interestingly, the value of the game $Wheel(n)$ does not depend on the option involving the removal of a rim edge. Upon further analysis of the values for pizzas, we noticed that the pizza sub-position that corresponds to the first move from a wheel which admits a value of 1 is also 0. Also there were some values of $Pizza(n-2)$ not equal to $Path(n-1)$ but there is no obvious pattern to their mismatch. This was surprising and would be interesting

to study further. However, the number of calculations required (and memory required) for large pizzas is unmanageable irrespective of the language used to analyze sub-positions. We coded using Python in SAGE [27], and were able to calculate up to $Wheel(98)$ which gives $Pizza(96)$. However, this did not allow us to determine the value for all wheels. Instead we analyzed the sub-positions combinatorially and noticed that pizzas do not significantly contribute to the mex calculation and thus can be ignored. Since RAM and time constraints contributed to the inability to analyze values for large wheels via a simple recursive code, it is unrealistic to expect pizzas to be easily computed.

2.2 Triple Packing Game

We now present a game which, to our knowledge, has not yet been analyzed. In order to understand the mathematical significance of the game, we require some background from design theory.

2.2.1 Design Theory

The definitions from this section are from [29] unless otherwise stated.

Design theorists are interested in studying all types of discrete structures based off of point sets and subsets of those point sets. Determining necessary and sufficient conditions for existence are of primary importance.

Formally, a *design* is a pair $(X, \mathcal{A})$ where X is a set of elements called points and $\mathcal{A}$ is a collection of non-empty subsets of X called blocks.

Imposing some structure on the design lends itself to nice applications. For example, suppose you had a set of students who will work with every other classmate exactly once throughout the term, but the classmates are always working in groups of three. Is it possible? How would this problem be solved mathematically? One could use designs which are a special

case of Balanced Incomplete Block Designs.

Let v, k and λ be positive integers where $v > k \geq 2$. A (v, k, λ)-*Balanced Incomplete Block Design* is a design $(X, \mathcal{A})$ such that X has v elements, each block is of size k, and every pair of distinct points is contained in exactly λ blocks.

Given the parameters on the designs, there are also restrictions on the number of blocks, b, appearing in $\mathcal{A}$, and the number of times a specific point appears in the block sets, r (point repetition number).

Theorem 14. [29] *In a (v, k, λ)-BIBD, every point occurs in exactly*

$$r = \frac{\lambda (v - 1)}{k - 1}$$

blocks.

Theorem 15. [29] *A (v, k, λ)-BIBD has exactly*

$$b = \frac{vr}{k} = \frac{\lambda (v^2 - v)}{k^2 - k}$$

blocks.

A (v, k, t)-*Steiner system* is an ordered pair $(X, \mathcal{A})$ such that the following properties are satisfied: X is a set of v elements called points, $\mathcal{A}$ is a collection of nonempty subsets of size k of X called blocks, and any t points are on a unique block. Relaxing the conditions slightly we obtain a different system: A (v, k, t)-*packing* is an ordered pair $(X, \mathcal{A})$ in which X is a set of v elements called points, every block $A \in \mathcal{A}$ is of size k and no t-subset of points is contained in more than one block. These relaxed conditions allow for some t-subsets to not be covered.

Example 10. *Let $X = \{1, 2, 3, 4, 5, 6, 7\}$ and let $\mathcal{A} = \{123, 145, 167, 246, 257, 347, 356\}$. This corresponds to the Fano plane pictured in Figure 2.2.*

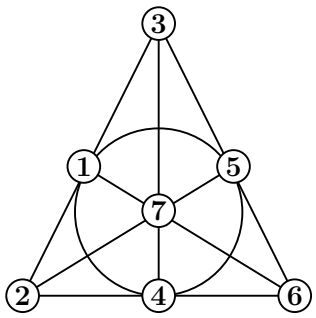

Figure 2.2: Fano Plane.

If the Steiner system has blocks of size three we call it a Steiner Triple System of order v. A *Steiner Triple System of order v (STS(v))* is an ordered pair $(X, \mathcal{A})$ with the following properties: X is a v-element set of points, $\mathcal{A}$ is a collection of 3-element subsets of X called triples (or blocks), and every pair of points is contained in exactly one triple.

Steiner triple systems only exist for certain sizes of v.

Theorem 16. [29] *There exists an STS(v) only if* $v \equiv 1, 3 \pmod 6$, $v \geq 7$.

Proof. Since this is a Steiner Triple System, which is also a $(v, 3, 1)$-BIBD, we know $k = 3$ and $\lambda = 1$. We simplify the formulas from Theorem 14 and Theorem 15, as both are in terms of v. First,

$$r = \frac{\lambda(v-1)}{k-1} = \frac{1(v-1)}{3-1} = \frac{v-1}{2}.$$

This implies (after rearranging) that $v = 2r + 1$. And hence, v is odd.

Secondly,

$$b = \frac{vr}{k} = \frac{\lambda(v^2-v)}{k^2-k} = \frac{1(v^2-v)}{3^2-3} = \frac{v^2-v}{6}.$$

This implies (after rearranging) that $v(v-1) = 6b$. Furthermore, $v(v-1) \equiv 0 \pmod 6$. Checking each value from the set $\{0, 1, 2, 3, 4, 5\}$ we find that $\{0, 1, 3, 4\}$ satisfy the equivalence. This, together with the fact that v is odd, gives that $v \equiv 1, 3 \pmod 6$. Lastly, since $v > k$ and $k = 3$, then $v \geq 7$. □

v $\equiv$	Maximum number of triples in packing	leave graph
1 or 3 (mod 6)	$\frac{v^2-v}{6}$	empty
0 or 2 (mod 6)	$\frac{v^2-2v}{6}$	$\frac{v}{2}K_2$
4 (mod 6)	$\frac{v^2-2v-2}{6}$	$K_{1,3} \cup \frac{v-4}{2}K_2$
5 (mod 6)	$\frac{v^2-v-8}{6}$	C_4

Table 2.8: Characterization of $2-(v,3,1)$-packings [9].

Other applications are not as strict as that presented at the beginning of this section. In some cases, we are interested in achieving the best possible number of subsets given certain constraints. This is an instance where maximal set systems are important. By Theorem 16, we know that maximal set systems may not be Steiner Triple Systems. Consider $v = 8$, what could a system look like? These special systems are called Steiner Triple Packings. A *Steiner Triple Packing* of order v is a set system $(X, \mathcal{A})$ where $|X| = v$, every block has size three, and every pair of points from X is contained in at most one block, A of $\mathcal{A}$. If there are no more triples which could be added to $\mathcal{A}$ we call the triple packing *maximal*. The sets $\{012, 034, 056, 135, 147, 246, 237\}$ and $\{012, 034, 056, 135, 147, 246, 257, 367\}$ are two maximal Steiner Triple Packings of order 8 of differing parity.

The *leave graph* is the graph remaining after the edges corresponding to the maximal triple packing have been removed from K_v. There is a nice characterization of the structure of optimal $2-(v,3,1)$-packings [9] in terms of their leave graphs (see Table 2.8).

Now that we have the necessary background, we can examine a triple packing game which was invented by Eric Mendelsohn in 1983 [26]. After an unsuccessful attempt to get the game published, it remained dormant for 32 years until now when we analyze it.

Game 7. *Consider a set X, where $|X| = v$, and the game begins with two sets: T, which consists of all the triples from the set X and an empty set $\mathcal{A}$. On their turn, players remove a triple from T and place it in $\mathcal{A}$ under the restriction that it cannot share a pair with any other triple already in $\mathcal{A}$. The last player to move wins.*

The end state of the triple packing game is that $\mathcal{A}$ is a maximal Steiner Triple Packing.

Example 11. *Let* $X = \{0, 1, 2, 3, 4\}$. *Then* $T = \{012, 013, 014, 023, 024, 034, 123, 124, 134, 234\}$. *One instance of the game play is as follows:*

$$T = \{012, 013, 014, 023, 024, 034, 123, 124, 134, 234\}, \mathcal{A} = \emptyset \tag{2.1}$$

$$T = \{034, 134, 234\}, \mathcal{A} = \{012\} \tag{2.2}$$

$$T = \emptyset, \mathcal{A} = \{012, 034\} \tag{2.3}$$

The game ends because there are no more admissible triples left in T. *The second player wins.*

To calculate the value of the game $\mathcal{G}$ we need to calculate its entire game tree and compute the values of every node. This is pictured in Figure 2.3. For notational simplification, the value of each node is given in parentheses and leaves with the same labels have been amalgamated.

In Figure 2.3, we see that the second player will always win in the game on 5 points since the value of $\mathcal{G}$ is 0. If we consider relabellings of values in X we can significantly reduce the search space. The first player can choose any triple from the set T but we can always relabel that triple to be 012 (the effect is to permute the elements of X). Hence all starting options for the next player are isomorphic. Without loss of generality, we can suppose we start with 012. This automatically prunes our search space by $\binom{|X|}{3} - 1 = \frac{|X|!}{3!(|X|-3)!} - 1$ initial branches. This observation leads us to another important conclusion about the game:

Theorem 17. *The value of the triple packing game is binary.*

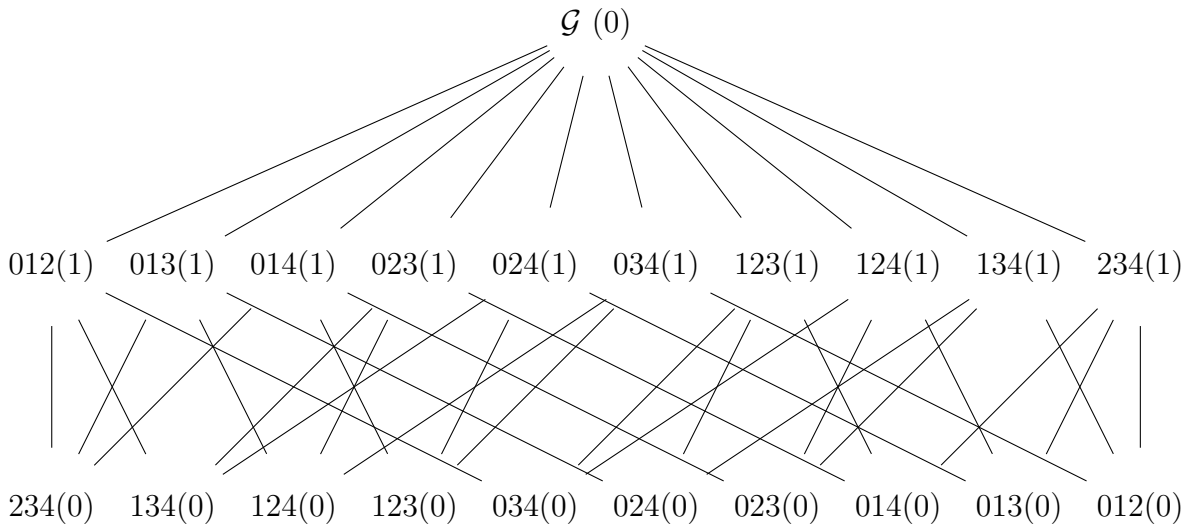

Figure 2.3: Game tree representation used to calculate the value of $\mathcal{G}$ on 5 points.

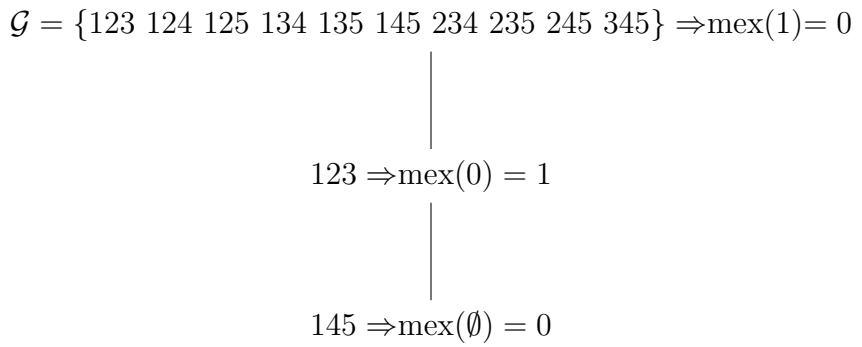

Figure 2.4: Isomorph free game tree for $|X| = 5$ with Grundy values.

Proof. By definition of mex, the value of the game $\mathcal{G}$ is the minimum excluded value for the set of options of $\mathcal{G}$. But we know that we can fix the first triple by relabelling. So the value of $\mathcal{G}$ is always based off of the mex of a single node option, 012. This option can be any value but given that we calculate the mex as the smallest non-negative integer not in the set of $\mathcal{G}$'s options, the set of values at that node cannot include both 0 and 1 simultaneously, and so the value of $\mathcal{G}$ must be 0 or 1 always. □

There are many different ways to analyze this game. Another equivalent interpretation is as follows. The game begins as a complete graph on n vertices, $G = K_n$. A move consists of choosing a triangle, K_3, from G and removing its edges. The game ends when the graph is triangle-free. This is the method we chose to code the Triple Packing game in Sage (see Algorithm 2 in Appendix A for our pseudocode). There are interesting programming simplifications. As mentioned earlier, we can always consider the first move to be 012 and hence removing this K_3 vastly reduces our search space. We can also more cleverly use the graph structure to further reduce the search space and break down the graph into simpler nim-sums. If there is a cut-vertex, we will treat each component as a summand in a disjunctive sum. For example, in Figure 2.5 the vertex labelled 'a' acts as a cut vertex. Instead of treating the graph as a whole, we consider the subgraphs 'abd' and 'ace' separately and nim-sum their values. In terms of storage of previously calculated nim-values of graphs, we are using the graph invariant of the size of the graph based on the number of edges within a graph. Two graphs cannot be isomorphic if they do not have the same number of edges. This simplification makes it faster to do a lookup in the dictionary. Also, a very important simplification we used was isomorph computation within SAGE which is based on Brendan McKay's NAUTY algorithm [25]. We compute a canonical labelling on the graph, and apply the isomorph check on incoming graphs in comparison to stored isomorphic certificates in the dictionary. If the graph has an isomorphic copy in the dictionary, we return the stored value. Otherwise, we compute it and store this new data. This significantly reduced the

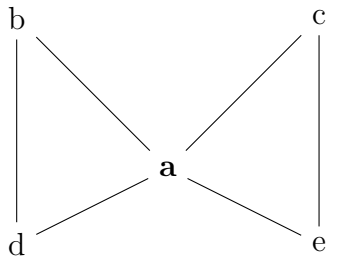

Figure 2.5: Cut vertex.

n	$\mathscr{G}(n)$	CPU time in seconds
0	0	-
1	0	-
2	0	-
3	1	0.00
4	1	0.01
5	0	0.02
6	0	0.10
7	1	0.30
8	1	1.07
9	0	3.13
10	0	34.65
11	1	976.12
12	1	90479.21

Table 2.9: The Triple Packing Game Nim-sequence and CPU time.

search space as there are many complex relabellings which can occur long after the first node.

Another way to think of this game on a graph is to consider the triples as vertices and two vertices are connected if the triples share a pair. This is called the Johnson Graph $J(n, 3)$ and the edges signify forbidden moves (for an example, see Figure 2.6). Interestingly, when we transform the triple packing game into a game played on the Johnson graph, the game is now Node-Kayles on $J(n, 3)$. Players alternate turns choosing a vertex, deleting it and its neighbours. The end state is a maximal independent set of vertices. Thus far, we have values for n from 0 to 12. The nim-sequence and CPU time is recorded in Table 2.9

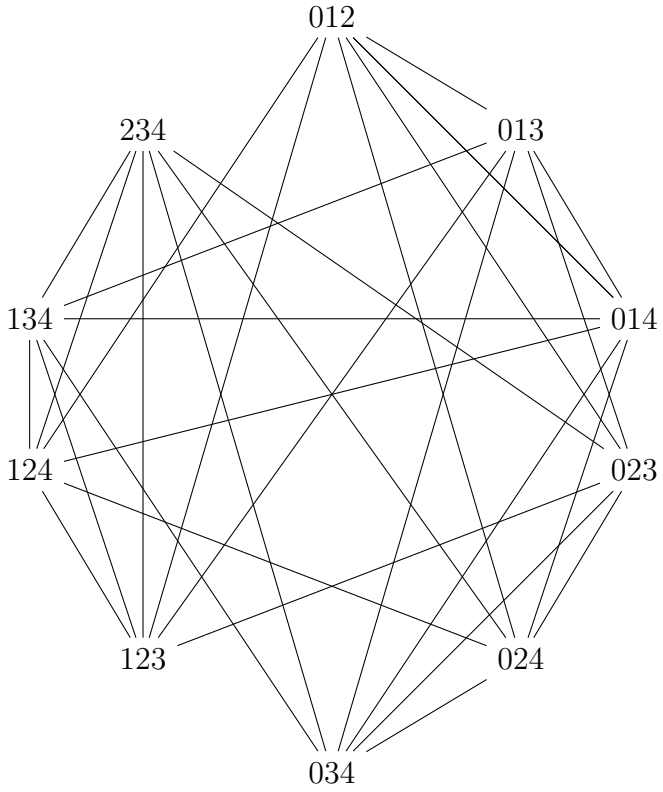

Figure 2.6: Johnson Graph, $J(5,3)$.

We believe the sequence continues with pre-period length 1, and period 0 0 1 1 . Unfortunately, even with dynamic programming and isomorph rejection, attaining large values of n proved to be very difficult. Recasting the Triple Packing game as Node-Kayles on the Johnson graph, $J(n, 3)$, leads us to believe that this problem is very difficult to solve since it was shown in 1998 that the independent set problem is NP-complete even in k-regular graphs [18] and recognizing when a graph has a decomposition into triangles is NP-complete [10]. At this time, we do not have enough empirical evidence to prove that the triple packing game sequence is even periodic.

Chapter 3

Future Directions

There are many different directions this work could be extended. We will now briefly explore a few of the possible directions for future work.

3.1 General future directions

3.1.1 A priori nim-value bounds and periodicity proof

It would be a great feat to develop methods of determining a priori bounds on complex impartial games. Also a priori understanding of periodic behaviour of games is important. Relying on ad-hoc methods for solving periodicity can lead to computational problems such as the problem we encountered with the wheel graph.

The game Subtraction(S) has bounds on all nim-values and as a consequence, has a nice classification for periodicity. Subtraction(S) is exactly the same as Nim except that instead of allowing the possibility to remove any i counters from a heap of size n, $1 \leq i \leq n$, a player can only remove the number of counters from the heap of size n which appear in a finite set S. If we could develop a priori bounds for our game, perhaps we would be able to emulate the proof for Subtraction(S) for Arc-Kayles on stars and also have a nice classification for

periodicity.

3.2 Arc-Kayles

3.2.1 Weights on vertices

A natural extension of this work is to consider weights placed on the vertices. The current set up is equivalent to having all weights being one. Imposing weights different than one on vertices allows for many different versions of the game to be studied. A few examples of such variants are enumerated here.

1. Decrease the number of counters by one on any pair of adjacent vertices. The edge remains after counter values are decreased.

2. Decrease the number of counters by one on any pair of adjacent vertices. Once you select a pair, the edge is removed.

3. Decrease the number of counters by $k \in \mathbb{N}$ on any pair of adjacent vertices with counter size l and m, such that $l \geq k$ and $m \geq k$. The edge remains after counter values are decreased.

4. Decrease the number of counters by $k \in \mathbb{N}$ on any pair of adjacent vertices with counter size l and m, such that $l \geq k$ and $m \geq k$. Once you select a pair, the edge is removed.

These games might be of interest for biological or city planning applications. For example, graphs with weights could model local protein-protein interactions or traffic congestion problems.

3.2.2 Different classes of graphs

It could be of interest to explore other classes of graphs. For example, imposing other restrictions on the graph such as triangle-free or k-regular.

3.2.3 Complexity

Computational complexity is an important area of study in computer science and mathematics. Understanding how hard a problem is can significantly impact what program to use to study a game, what memory allotment is necessary and how much time a project will take computationally. The complexity of determining the winner of Arc-Kayles, to date, is still unknown [28]. Interestingly, as mentioned earlier, Node-Kayles is P-SPACE complete, while the maximum independent set problem is NP-hard. Since, finding a maximum matching on an arbitrary graph is in P, we have the following conjecture.

Conjecture 2. *Arc-Kayles on an arbitrary graph can be solved in polynomial time.*

We are interested in exploring this as the next step of this research. We have begun the preliminary background understanding of this work, but have yet to obtain any results.

3.3 Triple Packing game

Solving the packing game in its entirety is, of course, the natural next step. We have the preliminary work set-up and thought out, three different ways to model the problem (triangle-free graph, set systems, Johnson graph) but have yet to have a nice understanding of the sub-positions as they grow exponentially with the value of n. Ultimately, understanding the Triple Packing Game on arbitrary hypergraphs is a primary end goal.

3.3.1 Complexity

As it is known that the problem of recognizing that a graph is the leave of a maximum triple packing is NP-complete [10], we suspect that the Triple Packing game on an arbitrary graph is also hard.

3.4 Variants on the design

We could extend the study of the intersection restriction games by changing the parameters on the design. For example, changing the design to be subsets of four points instead of three, increasing the value of λ to have larger intersection restrictions, or higher t. Before getting into these more complex intersection restrictions, it is necessary to fully understand the smallest non-trivial cases, Arc-Kayles and the Triple Packing game.

Appendix A

Pseudocode

Algorithm 1 mex(L)

P ← smallest non-negative integer not appearing in L
return P

Algorithm 2 TriplePackingGame()

```
Global Dictionary //Dictionary is a dictionary of previously visited graphs
External: mex()
List_of_connected_components ← list of connected components of G
One_connected_components ← [ ]
for G in List_of_connected_components do
  Identify cut vertices in G put each 1-connected component in the list of
  One_connected_components
end for
if length(List_of_one_connected_components) = 1 then
  G_canonical ← G.canonical_label()
  G_certificate ← G_canonical.certificate()
  num_verts ← G_canonical.order() //Determines the number of vertices in the graph G
  if graph and its value are in Dictionary then
    return value
  end if
  mex_list ← [ ]
  for edge in G_canonical do
    neighbours_1 ← G_canonical[edge[0]]//List of neighbours of the first vertex
    neighbours_2 ← G_canonical[edge[1]]//List of neighbours of the second vertex
    lex_larger_vertices ← [i for i in range(max(edge[0], edge[1]+1, num_verts))]
    list ← set_1.intersection_update(set_2, set_3) //Takes the intersection of all three sets
    for vertex in list do
      copyG ← deepcopy(G_canonical) //local copy
      copyG \K_3 (list) //Iterates possible triangles
      value ← TriplePackingGame(copyG)
      mex_list.append(value)
    end for
  end for
  Store G_canonical and nim_value in Dictionary
  return mex(nim_value)
else
  nim_sum ← 0
  for component in One_connected_components do
    nim_sum ⊕ = TriplePackingGame(component)
  end for
  return mex(nim_sum)
end if
```

Bibliography

[1] M. H. Albert, R. J. Nowakowski and D. Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory*. MA: A K Peters, Ltd. 2007.

[2] C. Berge. Two theorems in graph theory. Proceedings of the National Academy of Sciences of the United States of America. **43 (9)**, (1957), 842 – 844.

[3] E. R. Berlekamp, J. H. Conway, A. S. Fraenkel, R. J. Nowakowski, and V. Pless. *Combinatorial Games*. R. K .Guy, ed. **43**, Proceedings of symposia in applied Mathematics. Providence, RI: American Mathematical Society. 1991.

[4] E. R. Berlekamp, J. H. Conway and R. K. Guy. *Winning ways for your mathematical plays*. **1**, (2nd ed.), MA: A K Peters, Ltd. 2001.

[5] H. L. Bodlaender and D. Kratsch. Kayles and Nimbers. *Journal of Algorithms*. **43 (1)**, (2002), 106 – 119.

[6] H. L. Bodlaender, D. Kratsch and S. T. Timmer. Exact Algorithms for Kayles. *Theoretical Computer Science*. **562**, (2015), 165 – 176.

[7] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer Graduate Texts in Mathematics. New York. 2008.

[8] C. L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*. **3 (2)**, (1902), 35 – 39.

[9] C. J. Colbourn and J. H. Dinitz (eds.). *CRC Handbook of Combinatorial Designs*. FL: CRC Press. 1996.

[10] C. J. Colbourn and A. Rosa. *Triple Systems*. Oxford Mathematical Monographs. Oxford: Clarendon Press. 1999.

[11] J. H. Conway. *On Numbers and Games*. (Second Edition). MA: A K Peters, Ltd. 2001.

[12] M. Demange and T. Ekim. Efficient recognition of equimatchable graphs. *Information Processing Letters*. **114**, (2014), 66 − 71.

[13] E. Duchêne and G. Renault. Vertex Nim played on graphs. *Theoretical Computer Science*. **516**, (2014), 20 − 27.

[14] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*. **17**, (1965), 449 − 467.

[15] O. Favaron. Equimatchable factor-critical graphs. *Journal of Graph Theory*. **10**, (1986), 439 − 448.

[16] R. Fleischer and G. Trippen. Kayles on the way to the stars, in: H. J. van den Herik, Y. Björnsson, N. S. Netanyahu (Eds.), *Proceedings 4th International Conference on Computers and Games in: Lecture Notes in Computer Science*. **3846**, (July 2004), 232 − 245.

[17] A. Frendrup, B. Hartnell, and P. D. Vestergaard. A note on equimatchable graphs. *Australasian Journal of Combinatorics*. **46**, (2010), 185 − 190.

[18] G. H. Fricke, S. T. Hedetniemi, and D. P. Jacobs. Independence and irredundance in k-regular graphs. *ARS Combinatoria*. **49**, (1998), 271 − 279.

[19] M. Fukuyama. A Nim game played on graphs. *Theoretical Computer Science*. **304 (1)**, (2003), 387 – 399.

[20] A. Guignard and E. Sopena. Compound Node-Kayles on paths. *Theoretical Computer Science*. **410**, (2009), 2033 – 2044.

[21] P. Harding and P. Ottaway. Edge deletion games with parity rules. *Integers* [electronic version]. (2013). http://www.emis.de/journals/INTEGERS/papers/og1/og1.pdf

[22] K. Kawarabayashi, M. Plummer, and A. Saito. On two equimatchable graph classes. *Discrete Mathematics*. **266**, (2003), 263 – 274.

[23] M. Lesk, M. D. Plummer, and W. R. Pulleyblank. Equi-matchable graphs. *Graph theory and combinatorics*. London: Academic Press, 239 – 254, 1984.

[24] L. Lovász. *Combinatorial problems and exercises*. North Holland, Amsterdam. 1979.

[25] B. D. McKay and A. Piperno. nauty and Traces User's guide. Version 2.5. http://cs.anu.edu.au/ bdm/nauty/nug25.pdf.

[26] E. Mendelsohn. Personal Communication. (2015).

[27] SAGE Mathematics Software, Version 5.10, http://www.sagemath.org/.

[28] T. J. Schaefer. On the complexity of some two-person perfect information games. *Journal of computer and system sciences*. **16**, (1978), 185 – 225.

[29] D. R. Stinson. *Combinatorial Designs: Construction and Analysis*. New York: Springer-Verlag. 2004.

[30] D. P. Sumner. Randomly Matchable Graphs. *Journal of Graph Theory*. **3**, (1979), 183 – 186.