Enumeration, Isomorphism and Hamiltonicity of Cayley Graphs: 2-Generated and Cubic

by

Scott Effler

Bachelor of Science, University of Victoria, 2000

A Thesis Submitted in Partial Fullfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

We accept this thesis as conforming
to the required standard

Dr. Frank Ruskey, Supervisor

Dr. John Ellis, Departmental Member

Dr. Aaron Gulliver, Outside Member

Dr. Donovan Hare, External Examiner

Abstract

This thesis explores 2-generated and cubic Cayley graphs. All 2-generated Cayley graphs with generators from $\mathbb{S}_n$, where $n \leq 9$, were generated. Further, 3-generated cubic Cayley graphs, where $n \leq 7$, were also generated. Among these, the cubic Cayley graphs with up to 40320 vertices were tested for various properties including Hamiltonicity and diameter. These results are available on the internet in easy to read tables. The motivation for the testing of Cayley graphs for Hamiltonicity was the conjecture that states that every connected Cayley graph is Hamiltonian.

New enumeration results are presented for various classes of 2-generated Cayley graphs. Previously known enumeration results are presented for cubic Cayley graphs.

Finally, isomorphism and color isomorphism of 2-generated and cubic Cayley graphs is explored. Numerous new results are presented.

All algorithms used in this thesis are explained in full.

Examiners:

Dr. Frank Ruskey, Supervisor

Dr. John Ellis, Departmental Member

Dr. Aaron Gulliver, Outside Member

Dr. Donovan Hare, External Examiner

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

First and foremost, I would like to thank my supervisor, Frank Ruskey, for some of the figures seen in this thesis and, more importantly, for leading me in the right direction for this research. He has been a good mentor and teacher during my graduate program.

I would also like to give thanks to Brendan McKay for allowing me to use both his Nauty isomorphism checker as well as his cubham and digham programs for determining Hamiltonicity.

Also, I would like to thank both Ian Shields and Alex and Philipp Hertel for their work finding Hamilton cycles for some of the difficult graphs for which cubham had trouble.

On a personal note, I would like to thank my family and friends, especially Julia, for all their support.

Last but not least, I would like to thank my thesis committee for all of their time: John Ellis, Aaron Gulliver, Donovan Hare and Frank Ruskey.

Chapter 1

Introduction

In this research, we generated and tested many Cayley graphs for various properties such as isomorphism and Hamiltonicity. This data can be found on the internet at

$$http://www.theory.csc.uvic.ca/\sim cos/cayley/.$$

The aforementioned website presents tables of cubic and 2-generated Cayley graphs with various interesting properties. A few well known results pertaining to the subject are also given online. The production of this website is the main contribution of this thesis.

The initial motivation of this research was a long standing famous conjecture [40].

Conjecture 1.0.1 *(T.D. Parsons and others) All undirected connected Cayley graphs are Hamiltonian.*

It is widely believed that a counterexample to this conjecture, if one exists, will have small degree. Therefore, we set out to generate, enumerate and check all cubic Cayley graphs with a manageable number of vertices for Hamiltonicity.

This thesis is organized as follows. This chapter continues with definitions and an explanation of some applications for which Cayley graphs are used. Chapter 2 discusses the methods used to generate Cayley graphs and presents enumeration results. Chapters 3 and 4 present results pertaining to graph isomorphism and graph color isomorphism respectively. Chapter 5 contains a discussion of the Hamiltonicity checking that was performed on cubic and 2-generated Cayley graphs.

1.1 Definitions

A *set* is a collection of distinct objects with a precise description that provides a way of deciding whether a given object is in it [39].

Define a *graph* G as a set $V(G)$ of *vertices* and a set $E(G)$ of unordered pairs of vertices called *edges*. Further, a *directed graph* or *digraph* G is a set $V(G)$ of vertices and a set $A(G)$ of ordered pairs of vertices called *arcs*. A vertex $v \in V(G)$ is *incident* to edge $e \in E(G)$ if $e = (u, v)$ or $e = (v, u)$ for some $u \in V(G)$. The *degree* of vertex v in graph G, denoted $d(v)$, is the number of edges incident to v. A graph is *cubic* if all vertices have degree 3. Further, the $in - degree$ of vertex v in digraph G, denoted $d^-(v)$, is the number of arcs of the form (u, v) where $u \in V(G)$. Similarly, the $out - degree$ of vertex v in digraph G, denoted $d^+(v)$, is the number of arcs of the form (v, u) where $u \in V(G)$. A digraph G is $(x, y) - diregular$ if $d^-(v) = x$ and $d^+(v) = y$, $\forall v \in V(G)$.

A *group* G is a set together with a binary operation $*$ that has the following properties [32].

1. For every $a, b \in G$, $a * b \in G$.
2. For every $a, b, c \in G$ we have $a * (b * c) = (a * b) * c$.
3. There is an identity element $I \in G$ such that $\forall x \in G$, $I * x = x * I = x$.
4. $\forall x \in G$, $\exists x^{-1} \in G$ such that $x * x^{-1} = x^{-1} * x = I$.

An *equivalence relation* on a set S is a relation R on S such that for all choices of distinct $x, y, z \in S$ three properties hold [39].

1. $(x, x) \in R$ (reflexive property).
2. $(x, y) \in R$ implies $(y, x) \in R$ (symmetric property).
3. $(x, y), (y, z) \in R$ implies $(x, z) \in R$ (transitive property).

Let X be a subset of group G. Then X is a *generating set* for G if every element $g \in G$ can be written as $g = x_1 x_2 \cdots x_k$ for some k where $x_i \in X$ for $1 \leq i \leq k$. We call each $x \in X$ a *generator*. Define a *directed Cayley graph* or *Cayley digraph*, denoted $\overrightarrow{Cay}(X : G)$, as a graph whose vertex set is G and where arcs leaving vertex $g \in G$ are of the form (g, gx) for each $x \in X$ [11]. We will say this is the Cayley digraph generated by X. An *undirected Cayley graph* or simply a *Cayley graph*, denoted $Cay(X : G)$, has vertex set G and edges incident to $g \in G$ have the form (g,gx) and (g,gx^{-1}) for each $x \in X$ [34]. We will say this is the Cayley graph generated by X. A Cayley graph where $|X| = k$ is called a k-generated Cayley graph. For example, a Cayley graph where $|X| = 2$, is called a 2-generated Cayley graph.

Vertices u and v in graph G are *adjacent* if there exists edge $(u, v) \in E(G)$. A *path* from vertex u to vertex v is a sequence of adjacent vertices $u, x_1, x_2, \ldots, v$. A graph is *connected* if there exists a path from every vertex to every other vertex. In this research, we are only concerned with connected Cayley graphs. Therefore, for each generator set, we only consider the connected component that includes the identity element.

Define a *cycle* as a sequence of adjacent vertices $v_1, v_2, \ldots, v_n$ such that $(v_1, v_n) \in E(G)$ and there are no repeated vertices. Notice that we can similarly define a *directed cycle.*

A function α from a set A to a set B is $one - to - one$ if $\alpha(a) = \alpha(b)$ implies $a = b$. A function α from a set A to a set B is *onto* B if $\forall b \in B, \exists a \in A$ such that $\alpha(a) = b$. A *permutation* π of a set $S = \{1, 2, \ldots, n\}$ is a function from S to S that is both one-to-one and onto [11]. In other words, a permutation is an arrangement of the elements of the set in some order [32]. The *one line permutation notation* is $\pi_1 \pi_2 \cdots \pi_n$ where $\pi_i = \pi(i)$ for $1 \leq i \leq n$. The *cycle permutation notation* is $(a_1 a_2 \cdots a_m) \cdots$ where $a_{i+1} = a_i \pi$ for $1 \leq i < m$ and $a_1 = a_m \pi$. The dots at the end indicate the possibility that we may not have exhausted the set in the process, in which

case we continue the process with an element not already involved in a cycle. We call $(a_1 a_2 \cdots a_m)$ an $m-cycle$. An *involution* is a permutation that is its own inverse.

We are mainly concerned with 2-generated and cubic Cayley graphs. Note that cubic Cayley graphs can be divided into two groups. Class 1 cubic Cayley graphs consist of 2-generated Cayley graphs where exactly one generator is an involution. On the other hand, class 2 cubic Cayley graphs are those 3-generated Cayley graphs where all generators are involutions.

In this research, we will only consider Cayley graphs such that group G is a subgroup of the group of all length n permutations, $\mathbb{S}_n$ (where the group operation is permutation multiplication). This simplification can be justified by Cayley's Theorem [13].

Theorem 1.1.1 *Given any finite group of order m, there exists a group of permutations of m elements isomorphic to the given group.*

Let ϕ be an isomorphism between the finite group of order m and the group of permutations. Then, given $Cay(\{\sigma, \tau, \ldots\})$, $Cay(\{\phi(\sigma), \phi(\tau), \ldots\})$ is the corresponding isomorphic Cayley graph where the generators are from $\mathbb{S}_n$.

If the group is not specified in the Cayley graph notation, we assume that we are dealing with the connected component that includes the identity of the Cayley graph generated by the given generator set. For example, $Cay(\{(123)\})$ is the 3-cycle whose vertices are $\{I, (123), (132)\} = A_3$, the alternating group on 3 elements.

1.2 Example

A simple example will illustrate the somewhat complex definition of a Cayley graph. Figure 1.1 shows a drawing of the Cayley digraph $\overrightarrow{Cay}(\{(12), (012)\} : \mathbb{S}_3)$. Note that the first generator is represented by blue arcs and the second generator is represented by red arcs. As a convention, we will draw an edge

(x, y) if and only if there exists arcs (x, y) and (y, x) of the same color. Also, we will label vertices with their corresponding one line permutation notation.

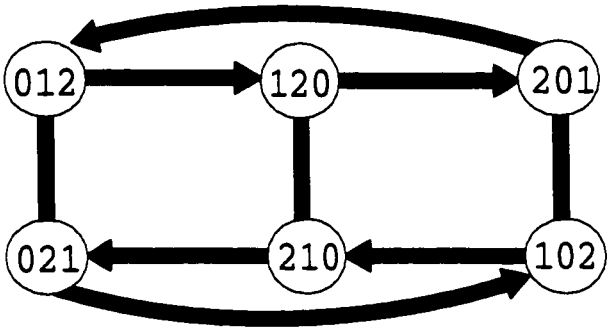


Figure 1.1: $\overrightarrow{Cay}(\{(12), (012)\} : \mathbb{S}_3)$

Essentially, we are applying the generators recursively to each vertex beginning with the identity until every vertex v satisfies $d^-(v) = d^+(v) = |X|$. For example, from vertex 210 we create arcs to $(12)210 = 120$ and $(012)210 = 021$.

1.3 Applications

Although Cayley graphs may seem very abstract, they have many useful applications.

1.3.1 Routing in a Network

In order for researchers to better understand networks, often they are first modeled as Cayley graphs. This way, the analysis often becomes more exact and it is sometimes easier to prove results.

Define an n-vertex graph G as a *generalized chordal ring* if vertices of G can be relabeled with integers such that vertex i is adjacent to vertex j if and only if vertex $i + q(mod\ n)$ is adjacent to vertex $j + q(mod\ n)$. Arden and Tang [3] showed that Cayley graphs can be represented as generalized chordal rings which have integer labels and are symmetrical. The vertices of a Cayley graph can thus be labeled with integers (modulo n) which is a tractable

addressing scheme for computer applications. Therefore, a straightforward routing algorithm based on table lookup is possible. Further, because these generalized chordal rings are symmetrical, they show that the routing table at each node is identical. This means that we need only store this table once, which saves much space over the naive routing algorithm.

Many other results have been published linking various subsets of Cayley graphs and networks [1, 2]. It is therefore evident that Cayley graphs help in the design and implementation of efficient networks.

1.3.2 Eccentric Digraphs

A second application of Cayley graphs is the exploration of the theory of eccentric digraphs. The *distance* from vertex u to vertex v in a graph is the length of the shortest path from u to v. The eccentricity $e(u)$ of vertex u is the maximum distance from u to any other vertex. We will say that a vertex v is an eccentric vertex of u if the distance from u to v is equal to $e(u)$. Further, the eccentric digraph $ED(G)$ of digraph G has the same vertex set as G and there exists an arc (u, v) if and only if v is an eccentric vertex of u [26].

The following lemma is derived from the definition of Eccentric digraphs.

Lemma 1.3.1 *The eccentric digraph of a Cayley digraph is also a Cayley digraph.*

Proof. New arcs in the eccentric digraph of a Cayley digraph correspond to paths in the original Cayley digraph. Each of these paths is in essence a sequence of generators, $g_1, g_2, \ldots, g_k$, from the original Cayley digraph. This implies a new generator, $g = g_1 g_2 \cdots g_k$, for each new arc. Since Cayley digraphs are vertex transitive (defined in section 3.3.2), this argument applies in the same way to each vertex (and creates the same generators for each vertex). Therefore, the eccentric digraph is also a Cayley digraph. □

Notice that this definition can be repeatedly iterated to yield the eccentric

digraph of an eccentric digraph and so on. Given a positive integer $k \geq 2$, the k^{th} iterated eccentric digraph of G is written as $ED^k(G)$. Therefore, an interesting question arises. What are the smallest integers $p > 0$ and $t \geq 0$ such that $ED^t(G) = ED^{p+t}(G)$? We call p the *period* of G and t the *tail* of G, denoted $p(G)$ and $t(G)$ respectively.

We calculated the period and the tail of the Cayley digraphs generated as part of this research and came across some interesting results. First, the great majority of the Cayley digraphs tested had a period of 2. However, at publication we had found 15 Cayley digraphs with a period of 4 as well as numerous Cayley digraphs with different periods. One of the Cayley graphs with a period of 4 is the undirected version of the Cayley graph pictured in Figure 1.2.

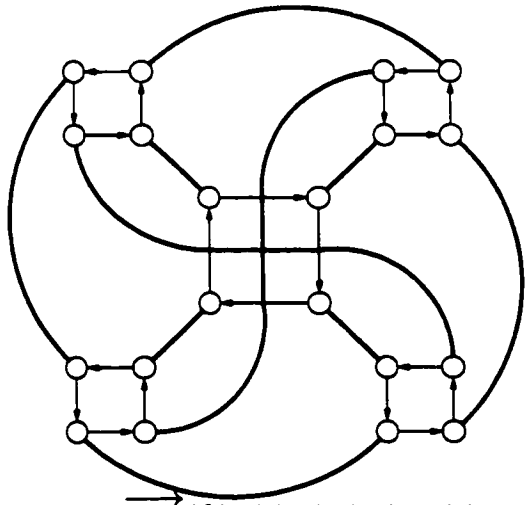


Figure 1.2: $\overrightarrow{Cay}(\{(56)(78), (01)(23)(4567)\})$

This is a research topic that is very much in its infancy. A paper of Miller, Gimbert, Ruskey and Ryan [26] gives a good survey of what is known and what is interesting in the area of eccentric digraphs. Testing for these properties on Cayley digraphs could prove to be a useful tool to aid in the understanding of eccentric digraphs.

1.3.3 Cellular Automaton

A third application of Cayley graphs is the modeling of cellular automata. One way to characterize how living cells act is to say that cells do not know their local situation with respect to their neighbors. Characterizations such as this one are used in a dissertation of Rocca [31]. This assumption allows us to model certain types of cellular automata as Cayley graphs and can greatly simplify the analysis of various properties of these cells.

1.3.4 Graph Restrictions

Sometimes, as every researcher knows, it is necessary to restrict the set we are working with in order for a stated conjecture to be true. Often, restricting the set of all graphs to the set of all Cayley graphs can be useful and may lead to an elegant proof.

As an example, we will consider a result of Babson and Benjamini [4]. Given two vertices u and v, define a (u, v) *cut set* as a set of edges that has nonempty intersection with every path from u to v. A *minimal cut set* is a (u, v) cut set that contains no proper set which is also a (u, v) cut set.

They show that various results about cut sets can be easily proved when the set of graphs is reduced to the set of Cayley graphs. For example, the quasi-connected minimal cut set property is known to hold for all finite Cayley graphs but does not hold for all general graphs [4].

1.3.5 Other Applications

Many other applications for Cayley graphs exist, some more important than others. Cayley graphs can be used to model designs [11], slider puzzles, bell ringing problems [13], economic theories [11], as well as many other interesting problems. Also, Cayley graphs are used extensively in many branches of mathematics and theoretical computer science and are interesting because of

their close correlation to group theory.

Chapter 2

Generation and Enumeration

A *Cayley pair* of order n is an unordered pair $\{\sigma, \tau\}$ such that $\sigma, \tau \in \mathbb{S}_n$. Cayley triples and Cayley k-sets are defined analogously. Further, notice that a Cayley pair is precisely the generating set for a 2-generated Cayley graph.

Cayley pair P_1 is *isomorphic* to Cayley pair P_2 if the symbols of P_1 can be relabeled to produce P_2. For example, Cayley pairs $\{(123), (23)\}$ and $\{(132), (12)\}$ are isomorphic since if we relabel the first Cayley pair $1 \mapsto 3$ and $3 \mapsto 1$ we obtain the second Cayley pair.

For each equivalence class of isomorphic Cayley pairs we define one special Cayley pair called the *canonic* representative. If we had a way to determine if a Cayley pair were a canonic representative of an equivalence class we could easily discard isomorphic Cayley pairs and generate only the non-isomorphic Cayley pairs. Note that the idea of canonicity can be applied to any group with distinct equivalence classes. We will examine how to determine whether a Cayley pair is canonic in a later section.

2.1 Enumeration of Cayley Pairs

Counting the number of Cayley pairs of order n, P_n, is trivial. If there were a small number of these pairs, we could use a brute force method to generate all of them.

First, the number of different generators is simply the number of permutations of an n-set. This quantity is $n!$. Therefore, P_n is the number of ways to choose two items from $n!$ items.

$$P_n = \binom{n!}{2} \tag{2.1}$$

Notice that this formula allows the identity permutation element to be a generator. Further, notice that the number of Cayley k-sets for a given value of n is clearly $\binom{n!}{k}$.

Values of this quantity are shown in Table 2.1. Notice that the values become large very quickly, thus, generating all pairs is not feasible. We must reject isomorphic Cayley pairs.

2.2 Enumeration of Cubic Cayley Triples

Further to counting Cayley pairs, we now count Cayley triples that generate class 2 cubic Cayley graphs. Again, if there were a small number of these triples, we could use brute force to generate all of them.

Define I_n as the number of permutations of length n that are involutions. Following Knuth [17], we derive a formula for I_n by noticing that the first element can either be contained in a 1-cycle or a 2-cycle. If it is involved in a 1-cycle then there are I_{n-1} valid ways to complete the permutation. If it is involved in a 2-cycle, there are $n-1$ possible elements to complete the cycle and for each there are I_{n-2} valid ways to complete the permutation.

$$I_n = (n-1)I_{n-2} + I_{n-1},\ I_0 = 0,\ I_1 = 1 \tag{2.2}$$

A closed formula is also known, but is not relevant to this research.

The number of Cayley triples of order n where all generators are involutions (class 2 cubic Cayley graphs) is $\binom{I_n}{3}$. Notice that the number of Cayley k-sets where all generators are involutions is $\binom{I_n}{k}$.

We could extend this result to obtain a formula for the number of Cayley pairs on n symbols with l involutions and k non-involutions as $\binom{I_n}{l}\binom{n!-I_n}{k}$. Using this, the number of Cayley pairs of order n where exactly one generator is an involution (class 1 cubic Cayley graphs) is $\binom{I_n}{1}\binom{n!-I_n}{1} = I_n(n! - I_n)$. Enumerations for $n \leq 8$ are shown in Table 2.1.

It is apparent, as it was for Cayley pairs, that there are far too many triples

	(Pairs)		(Cubic Pairs)	(Cubic Triples)
n	P_n	I_n	$I_n(n! - I_n)$	$\binom{I_n}{3}$
1	0	1	0	0
2	1	1	1	0
3	15	3	9	1
4	276	6	108	20
5	7140	18	1836	816
6	258840	48	32256	17296
7	12698280	156	761904	620620
8	812831040	492	19595376	19728380

Table 2.1: Enumerations of Cayley Pairs and Triples

to generate all of them. We must reject isomorphic Cayley triples.

2.3 Generation

Since there are too many Cayley pairs to store, we will test each possible Cayley pair to see if it is isomorphic to any other Cayley pair. Before we can do this, we must generate permutations of an n-set (the Cayley generators). To that end, Algorithm 4.11 (page 66) of Ruskey [32] was used in this research. This algorithm runs in constant amortized time [32] per permutation generated, which is always preferred.

2.3.1 Algorithm

Relative to an ordering of a set of combinatorial objects, the *rank* of an object π, written $rank(\pi)$, is the position that the object occupies in the ordering [32]. To *unrank* an integer r, written $unrank(r)$, is to produce the object π where $rank(\pi) = r$ [32]. In this research, we will use lexicographic order (dictionary ordering) of the one line permutation notation.

We generate all non-isomorphic Cayley pairs by testing for two simple

conditions.

First, we eliminate all Cayley pairs that have a common fixed point (1-cycle). These Cayley pairs can be eliminated since there will be an isomorphic Cayley pair with smaller n.

Second, we eliminate any Cayley pair whose labeling is not canonic. Let $k_\sigma = \{\pi\sigma\pi^{-1} | \pi \in \mathbb{S}_n\}$ be the *conjugacy class* of $\sigma \in \mathbb{S}_n$. As noted in exercise 3 (page 72) in Joseph Gallian's book [11], conjugation preserves structure. In other words, k_σ is the equivalence class of all permutations isomorphic to σ. Now, we define the canonic representative of k_σ to be the lexicographic smallest element from k_σ, when written in one line notation.

Note that we must apply the conjugacy class definition to each generator separately. Recall that Cayley pairs are unordered and therefore the elements derived from their corresponding conjugacy classes can be in either order.

Relative to an ordering of a set of combinatorial objects, we say $\{\sigma', \tau'\} < \{\sigma, \tau\}$ (assuming $\sigma < \tau$ and $\sigma' < \tau'$) if and only if $rank(\sigma') < rank(\sigma)$ or, $rank(\sigma') = rank(\sigma)$ and $rank(\tau') < rank(\tau)$. In our implementation, we used lexicographic order of the one line permutation notation. Algorithm 2.1 generates all non-isomorphic Cayley pairs.

2.3.2 Example

As an example, we will consider generating all non-isomorphic Cayley pairs for $n = 3$. First, notice that there are 6 possible generators when $n = 3$, listed on elements {0,1,2}.

$$(0)(12),\ (02)(1),\ (01)(2),\ (012),\ (021),\ (0)(1)(2)$$

Further, there are $\binom{6}{2} = 15$ Cayley pairs. A subset of these is listed below.

$$\{(12),(01)\}, \{(12),(012)\}, \{(02),(01)\}$$

Among these, {(12),(01)} and {(12),(012)} will be generated by Algo-

Given integer n

begin

  for ***each*** $\{\sigma, \tau\}$ *where* $\sigma, \tau \in \mathbb{S}_n$ *and* $\sigma < \tau$ **do**

    if *isCannoc*$(\{\sigma, \tau\})$ *and there is no common fixed point* **then**

      Output Cayley pair

    endif

  endfor

end

isCannoc$(\{\sigma, \tau\})$ **begin**

  for ***each*** $\pi \in \mathbb{S}_n$ **do**

    $\sigma' = \pi\sigma\pi^{-1}$; $\tau' = \pi\tau\pi^{-1}$;

    if $\{\sigma', \tau'\} < \{\sigma, \tau\}$ *OR* $\{\tau', \sigma'\} < \{\sigma, \tau\}$ **then**

      return FALSE;

    endif

  endfor

  return TRUE;

end

Algorithm 2.1: Generating Non-Isomorphic Cayley Pairs

rithm 2.1. It turns out that the third Cayley pair listed above will not be generated by the algorithm. To prove this using the method of Algorithm 2.1, consider $\pi = (021)$. This implies $\pi^{-1} = (012)$.

$$\pi(02)\pi^{-1} = (01) = 102$$

$$\pi(01)\pi^{-1} = (12) = 021$$

Since $\{021, 102\} < \{102, 210\}$ this pair is not canonic and is therefore not generated. Notice that in the $n = 3$ case some pairs are eliminated because they have a common fixed point with another pair. For example, we eliminate {(01)(2),(0)(1)(2)} since this pair is isomorphic to {(01),(0)(1)} in $n = 2$. The common fixed point is element 2.

2.3.3 Correctness

A simple argument based on the definition of a conjugacy class proves correctness.

Theorem 2.3.1 *Algorithm 2.1 generates all non-isomorphic Cayley pairs for a given n.*

Proof. The algorithm tests each possible Cayley pair for canonicity. For a given Cayley pair we compute the associated elements from the conjugacy class for each generator. The lexicographic smallest unordered pair of elements from the conjugacy classes of each generator is lexicographically less than the original unordered Cayley pair if and only if the Cayley pair is not canonic. Due to the fact that conjugation preserves structure (i.e. isomorphism) and only one pair in each class can be the lexicographic least, this method returns FALSE for all but one of the Cayley pairs in an equivalence class of isomorphic Cayley pairs. □

2.3.4 Extension to Cayley Triples

Algorithm 2.2 was used to generate all cubic Cayley triples. It is an extension of Algorithm 2.1. We say $\{\sigma', \tau', \rho'\} < \{\sigma, \tau, \rho\}$ (assuming $\sigma < \tau < \rho$ and $\sigma' < \tau' < \rho'$) if and only if $rank(\sigma') < rank(\sigma)$ or, $rank(\sigma') = rank(\sigma)$ and $rank(\tau') < rank(\tau)$ or, $rank(\sigma') = rank(\sigma)$ and $rank(\tau') = rank(\tau)$ and $rank(\rho') < rank(\rho)$.

2.4 Adjacency Lists

Converting Cayley pairs into their corresponding adjacency list representation is straight forward.

```
Given integer n
begin
   for each {σ, τ, ρ} where σ, τ, ρ ∈ S_n and σ < τ < ρ do
      if isCannoc({σ, τ, ρ}) and there is no common fixed point then
         Output Cayley triple
      endif
   endfor
end
isCannoc({σ, τ, ρ}) begin
   for each π ∈ S_n do
      σ' = πσπ^-1; τ' = πτπ^-1; ρ' = πρπ^-1;
      Reorder {σ', τ', ρ'} such that σ' < τ' < ρ'
      if {σ', τ', ρ'} < {σ, τ, ρ} then
         return FALSE;
      endif
   endfor
   return TRUE;
end
```

Algorithm 2.2: Generating Non-Isomorphic Cayley Triples

2.4.1 Algorithm

As in the definition of a Cayley graph, we simply apply each generator to every vertex starting at the identity. Notice that ranking and unranking algorithms are needed. Our method will work with any ranking and unranking algorithms as long as they both use the same permutation ordering. In our implementation, we used the linear time ranking and unranking of Myrvold and Ruskey [28]. Algorithm 2.3 converts Cayley pairs into adjacency lists.

Note that this algorithm will generate $|\mathbb{S}_n|$-vertex graphs. We are really only concerned with a connected component of this graph.

This algorithm could be easily converted to a recursive version. At each

Given Cayley pair $\{\sigma, \tau\}$
begin
  for $i = 0$ *to* $|\mathbb{S}_n| - 1$ **do**
    π = unrank(i);
    $\sigma' = \sigma\pi$; $\tau' = \tau\pi$;
    add arcs $(i, rank(\sigma'))$ and $(i, rank(\tau'))$
  endfor
end

Algorithm 2.3: Converting Cayley Pairs to Adjacency Lists

step, add arcs corresponding to the current vertex, then call the function on new vertices reached by the arcs. We are finished when there are no more arcs to create (i.e., $d(v)^+ = 2, \forall v \in V(G)$).

2.4.2 Example

As an illustration, consider $\overrightarrow{Cay}(\{(12), (012)\} : \mathbb{S}_3)$ as drawn in Figure 1.1. Algorithm 2.3 applies the generators starting from the identity (assume $rank(012) = 0$). For example, let $\pi = unrank(0) = 012$. Then, $\sigma' = (12)\pi = 021$ and $\tau' = (012)\pi = 120$. Therefore, we add arcs $(0, rank(021))$ and $(0, rank(120))$. This is continued until there are no more arcs to create. Notice that there should be two arcs leaving each vertex. When this is achieved, we are done.

2.4.3 Extension to Cayley Triples

Converting Cayley triples to adjacency lists is a simple extension of Algorithm 2.3 and is shown as Algorithm 2.4

2.5 Data

We will begin by defining some symbols.

- $I_{2,n}$ is the number of non-isomorphic Cayley pairs on n symbols. For

Given Cayley triple $\{\sigma, \tau, \rho\}$
begin
  for $i = 0$ *to* $|\mathbb{S}_n| - 1$ **do**
    π = unrank(i);
    $\sigma' = \sigma\pi$; $\tau' = \tau\pi$; $\rho' = \rho\pi$;
    add arcs $(i, rank(\sigma'))$, $(i, rank(\tau'))$ and $(i, rank(\rho'))$
  endfor
end

Algorithm 2.4: Converting Cayley Triples to Adjacency Lists

example, $I_{2,3} = 5$ since the five valid Cayley pairs are $\{(12), (0)(1)(2)\}$, $\{(12), (01)\}$, $\{(012), (021)\}$, $\{(12), (012)\}$ and $\{(012), (0)(1)(2)\}$

- $F_{2,n}$ is the number of non-isomorphic Cayley pairs with no common fixed point on n symbols. For example, $F_{2,3} = 4$ since the four valid Cayley pairs are $\{(12), (01)\}$, $\{(012), (021)\}$, $\{(12), (012)\}$ and $\{(012), (0)(1)(2)\}$

- $I'_{2,n}$ is the number of non-isomorphic Cayley pairs on n symbols where exactly one generator is an involution. For example, $I'_{2,3} = 2$ since the only two valid Cayley pairs are $\{(12), (012)\}$ and $\{(12), (0)(1)(2)\}$.

- $F'_{2,n}$ is the number of non-isomorphic Cayley pairs with no common fixed point on n symbols where exactly one generator is an involution. For example, $F'_{2,3} = 1$ since the only valid Cayley triple is $\{(12), (012)\}$.

- $I'_{3,n}$ is the number of non-isomorphic Cayley triples on n generator symbols where all generators are involutions. For example, $I'_{3,3} = 1$ since the only valid Cayley triple is $\{(01), (02), (12)\}$.

- $F'_{3,n}$ is the number of non-isomorphic Cayley triples that do not share a common fixed point on n generator symbols where all generators are involutions. For example, $F'_{3,3} = 1$ since the only valid Cayley triple is $\{(01), (02), (12)\}$.

The resulting Cayley graphs from $I'_{2,n}$, $F'_{2,n}$, $I'_{3,n}$ and $F'_{3,n}$ are cubic. We generalize these definitions to obtain $I_{k,n}$, $F_{k,n}$, $I'_{k,n}$, and $F'_{k,n}$.

Non-isomorphic Cayley pairs were generated exhaustively for $n \leq 8$ and non-isomorphic Cayley pairs where exactly one generator is an involution were generated for $n = 9$. Enumerations are recorded in Table 2.2.

n	$I_{2,n}$	$F_{2,n}$	$I'_{2,n}$	$F'_{2,n}$	$I'_{3,n}$	$F'_{3,n}$
1	0	0	0	0	0	0
2	1	0	1	0	0	0
3	5	4	2	1	1	1
4	23	18	9	7	10	9
5	89	66	26	17	37	27
6	484	395	98	72	197	160
7	2904	2420	270	172	676	479
8	22002	19098	913	643	3094	2418
9			2645	1732	12022	8928

Table 2.2: Enumerations of Non-Isomorphic Cayley Pairs and Triples From Data

2.6 Results

Some simple inequalities as well as more difficult results based on Burnside's Lemma were derived from this data.

2.6.1 Simple Formulae

Lemma 2.6.1 *$F_{k,n} \leq I_{k,n}$ and $F'_{k,n} \leq I'_{k,n}$ for $n \geq k \geq 2$.*

Proof. Obvious since we are further restricting the set of Cayley k-sets. □

Lemma 2.6.2 *$I_{k,n} = I_{k,n-1} + F_{k,n}$ and $I'_{k,n} = I'_{k,n-1} + F'_{k,n}$ for $n \geq k \geq 2$.*

Proof. Notice that a Cayley k-set on n symbols can be split into two classes; those sets that are fixed point free and those that are not. The number of Cayley k-sets on n symbols that are fixed point free is by definition $F_{k,n}$. Conversely, the number of Cayley k-sets on n symbols that are not fixed point free is precisely the number of Cayley k-sets on $n-1$ symbols (with a fixed point appended on the end). This quantity is $I_{k,n-1}$. Thus, we have our result. The same argument applies to Cayley k-sets with exactly one involution. □

For example, from our data $I_{2,5} + F_{2,6} = 89 + 395 = 484 = I_{2,6}$.

2.6.2 Enumeration of Non-Isomorphic Cayley Pairs

We will use an adaptation of Burnside's Lemma [7] to develop a formula for the number of non-isomorphic Cayley pairs with given n.

Theorem 2.6.3 *Let G be a subgroup of $\mathbb{S}_n$. The number of distinct orbits associated with G (i.e., the number of non-isomorphic Cayley pairs) is given by*

$$\frac{Fix(g_1) + Fix(g_2) + \cdots + Fix(g_{|G|})}{|G|}$$

where $g_i \in G$ and $Fix(g_i)$ is the number of elements $x \in G$ such that $xg_i = x$.

As an example of Theorem 2.6.3 we will compute the number of non-isomorphic Cayley pairs of order 3.

(0)(1)(2) leaves $\binom{3!}{2} = 15$ Cayley pairs fixed (i.e., all of them). (0)(12) leaves {(0)(1)(2),(0)(12)}, {(012),(021)} and {(1)(02),(2)(01)} fixed. (1)(02), (2)(01), (012) and (021) similarly each leave 3 pairs fixed. Therefore, the number of non-isomorphic Cayley pairs is

$$\frac{15+3+3+3+3+3}{6} = 5.$$

A *numerical partition* of a positive integer n is a sequence $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0$ such that $\lambda_1 + \lambda_2 + \cdots + \lambda_k = n$. Numerical partitions of n are denoted

by $p(n)$. If we rewrite the above sum by grouping together all permutations that have the same cycle structure, it is obvious that these cycle structures are exactly the numerical partitions of n.

$$\frac{1 \cdot 15 + 3 \cdot 3 + 2 \cdot 3}{3!} = 5$$

Let, $\boldsymbol{\lambda} = \lambda_1, \lambda_2, \ldots, \lambda_n$, then

- $P(\boldsymbol{\lambda})$ is the set of permutations of n with λ_i i-cycles for $1 \leq i \leq n$.
- $P_{1,n}(\boldsymbol{\lambda}) = |\{\pi \in \mathbb{S}_n : \sigma(\pi) = \pi\}|$ where $\sigma \in P(\boldsymbol{\lambda})$.
- $P_{2,n}(\boldsymbol{\lambda}) = |\{\pi \in \mathbb{S}_n : \sigma^2(\pi) = \pi \wedge \sigma(\pi) \neq \pi\}|$ where $\sigma \in P(\boldsymbol{\lambda})$.

Further, let $\boldsymbol{\lambda}^2$ be the cycle structure of the permutation that results from multiplying a permutation with cycle structure $\boldsymbol{\lambda}$ by itself.

We will now apply Theorem 2.6.3 to Cayley pairs in order to compute $I_{2,n}$. Notice that to fix a pair, either both permutations remain fixed, or they map to each other. We now see a formula arise.

$$n! \cdot I_{2,n} = \sum_{\boldsymbol{\lambda} \in p(n)} \left[|P(\boldsymbol{\lambda})| \left(\binom{P_{1,n}(\boldsymbol{\lambda})}{2} + \frac{P_{2,n}(\boldsymbol{\lambda})}{2} \right) \right] \tag{2.3}$$

These quantities have formulae for them that will simplify our equation. The Cauchy formula [7] is useful.

Lemma 2.6.4 $|P(\boldsymbol{\lambda})| = \frac{n!}{\lambda_1! \lambda_2! \ldots \lambda_n! 1^{\lambda_1} 2^{\lambda_2} \cdots n^{\lambda_n}}$

A result similar to the Cauchy formula is also necessary.

Lemma 2.6.5 $P_{1,n}(\boldsymbol{\lambda}) = \lambda_1! \lambda_2! \cdots \lambda_n! 1^{\lambda_1} 2^{\lambda_2} \cdots n^{\lambda_n}$

Proof. We will count the number of permutations π, such that $\sigma(\pi) = \pi$. First, notice that cycles of the same size can be "permuted" so that the entire permutation remains fixed. This can be done in $\lambda_1! \lambda_2! \cdots \lambda_n!$ ways. Second, notice that the cycles themselves can be rotated so that the entire permutation

remains fixed. This can be done in $1^{\lambda_1}2^{\lambda_2}\cdots n^{\lambda_n}$ ways. These are the only two possibilities and thus, we have shown the result. □

Finally, results about applying permutations with a certain cycle structure to themselves are needed.

Lemma 2.6.6 $P_{2,n}(\boldsymbol{\lambda}) = P_{1,n}(\boldsymbol{\lambda}^2)$ - $P_{1,n}(\boldsymbol{\lambda})$

Proof. By definition of $P_{2,n}(\boldsymbol{\lambda})$. □

Lemma 2.6.7 $P_{1,n}((\lambda_1, \lambda_2, \ldots, \lambda_n)^2) = P_{1,n}(\lambda'_1, \lambda'_2, \ldots, \lambda'_n)$

$$\textit{where } \lambda'_i = \begin{cases} 2\lambda_{2i} & \textit{if } i \textit{ is even} \\ \lambda_i + 2\lambda_{2i} & \textit{if } i \textit{ is odd} \end{cases}$$

Proof. Applying an odd cycle to itself, results in the same sized odd cycle. Applying an even cycle of length $2i$ to itself results in two length i cycles, one with every second element starting at the first element, and the other with every second element starting with the second element. The result follows. □

If we apply the above results to (2.3) we obtain the following formula.

$$\begin{aligned} n!\cdot I_{2,n} &= \sum_{\boldsymbol{\lambda}\in p(n)} \left[|P(\boldsymbol{\lambda})| \left(\binom{P_{1,n}(\boldsymbol{\lambda})}{2} + \frac{P_{2,n}(\boldsymbol{\lambda})}{2} \right) \right] \\ &= \sum_{\boldsymbol{\lambda}\in p(n)} \frac{n!}{\lambda_1!\cdots\lambda_n!1^{\lambda_1}\cdots n^{\lambda_n}} \\ &\qquad \left(\binom{\lambda_1!\cdots\lambda_n!1^{\lambda_1}\cdots n^{\lambda_n}}{2} + \frac{P_{1,n}((\boldsymbol{\lambda})^2) - \lambda_1!\cdots\lambda_n!1^{\lambda_1}\cdots n^{\lambda_n}}{2} \right) \\ I_{2,n} &= \sum_{\boldsymbol{\lambda}\in p(n)} \left(\frac{\lambda_1!\cdots\lambda_n!1^{\lambda_1}\cdots n^{\lambda_n}}{2} + \frac{\lambda'_1!\cdots\lambda'_n!1^{\lambda'_1}\cdots n^{\lambda'_n}}{2\lambda_1!\cdots\lambda_n!1^{\lambda_1}\cdots n^{\lambda_n}} - 1 \right) \end{aligned}$$

$$\textit{where } \lambda'_i = \begin{cases} 2\lambda_{2i} & \text{if } i \text{ is even} \\ \lambda_i + 2\lambda_{2i} & \text{if } i \text{ is odd} \end{cases} \tag{2.4}$$

Using (2.4), Lemma 2.6.2 and a numerical partition generation algorithm from [32] we implemented a C program to construct a table of values for $I_{2,n}$ and $F_{2,n}$. Table 2.3 contains these values for $n \leq 12$.

n	$I_{2,n}$	$F_{2,n}$
1	0	0
2	1	0
3	5	4
4	23	18
5	89	66
6	484	395
7	2904	2420
8	22002	19098
9	190555	168553
10	1876337	1685782
11	20445337	18569000
12	244087420	223642083

Table 2.3: Enumerations of Non-Isomorphic Cayley Pairs From Formula

Notice that the values computed by the formula match those found by exhaustive computer search as seen in Table 2.2.

2.6.3 Enumeration of Non-Isomorphic Cubic Cayley Pairs and Triples

Given a sequence of numbers $S = a_0, a_1, \ldots$, we define the expression

$$Z(x) = a_0 + a_1 x + a_2 x^2 + \cdots$$

to be the generating function of the sequence S [7].

The enumeration formula developed for Cayley pairs cannot be directly applied to $I'_{2,n}$ and $I'_{3,n}$ since these quantities count only pairs that generate cubic Cayley graphs. In other words, our isomorphism condition is slightly

different. We instead use an extension of a result by Curtin [9] which uses generating functions.

Let X_n be the set of equivalence classes in $\mathbb{S}_n$ under the isomorphism equivalence relation. Let $G.\boldsymbol{\lambda}$ denote group G regarded as a permutation group acting on the elements of X_n of cycle structure $\boldsymbol{\lambda}$. Let $PC(\alpha)$ be the pseudo-centralizer of permutation element α defined as follows.

$$PC(\alpha) = C(\alpha) \cup F(\alpha) \tag{2.5}$$

where

$$C(\alpha) = \{\beta \in \mathbb{S}_n | \alpha^\beta = \alpha\}$$
$$F(\alpha) = \{\beta \in \mathbb{S}_n | \alpha^\beta = \alpha^{-1}\}$$

Let $order(\boldsymbol{\lambda})$ be the smallest $n > 1$ such that $\boldsymbol{\lambda}^n = \boldsymbol{\lambda}$. Further, let $[t]Q$ be the coefficient of monomial t in polynomial Q and let $x_{(\boldsymbol{\lambda},k)}$ denote a k-cycle of elements of type $\boldsymbol{\lambda}$. Also, $\mu(n)$ is the *Möbius* function defined as follows.

$$\mu(n) = \begin{cases} (-1)^t & \text{if } n \text{ is the product of } t \text{ distinct prime numbers} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Now, we state and sketch a proof of Curtin's main theorem.

Theorem 2.6.8 *The generating function for enumerating Cayley sets, Z, is as follows:*

$$Z(\mathbb{S}_n.X_n) = \frac{1}{n!} \sum_{\boldsymbol{\lambda} \in p(n)} |P(\boldsymbol{\lambda})| m(\boldsymbol{\lambda}, X_n) \tag{2.7}$$

where

$$m(\boldsymbol{\lambda}, X_n) = \prod_{\boldsymbol{\lambda}' \in p(n)} m(\boldsymbol{\lambda}, \boldsymbol{\lambda}')$$
$$m(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \prod_{k | order(\boldsymbol{\lambda})} x_{(\boldsymbol{\lambda}',k)}^{c(\boldsymbol{\lambda},k,\boldsymbol{\lambda}')}$$
$$c(\boldsymbol{\lambda}, k, \boldsymbol{\lambda}') = \frac{1}{k} \sum_{d|k} \mu(k/d) w(\boldsymbol{\lambda}^d, \boldsymbol{\lambda}')$$
$$w(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \frac{n!}{|P(\boldsymbol{\lambda})|} [\boldsymbol{\lambda}] Z(PC(\boldsymbol{\lambda}'))$$

Now, let $F_n(x,y)$ be the polynomial obtained from $Z(\mathbb{S}_n..X_n)$ by replacing $x_{(\lambda,k)}$ by $(1+x^k)$ if $order(\lambda) \geq 3$, $(1+y^k)$ if $order(\lambda) = 2$ and 1 otherwise.

When expanded, the $x^r y^s$ term of this generating function is the number of non-isomorphic Cayley $(2r+s)$-sets where s generators are involutions. These enumerations do not include the identity permutation.

Proof. Here we give a sketch of the proof given by Curtin [9].

Let $f(\alpha_1, \lambda') = |\{\beta \in \lambda' | \beta^{\alpha_1} = \beta \text{ or } \beta^{-1}\}|$. By some simple algebra, for any $\beta_1 \in \lambda'$, we get

$$|P(\lambda)| f(\alpha_1, \lambda') = |P(\lambda')| [\lambda] Z(PC(\beta_1)) \tag{2.8}$$

Let $w(\alpha, \lambda')$ denote the number of elements of X_n of cycle type λ' fixed under conjugation. Substituting into (2.8) results in

$$w(\alpha, \lambda') = \frac{n!}{|P(\lambda)|} [\lambda] Z(PC(\lambda')) \tag{2.9}$$

Now, $c(\lambda, k, \lambda')$ is the number of k-cycles of the action of any α of cycle type λ by conjugation on the elements of X_n of cycle type λ'. Having said this, the equations of Theorem 2.6.8 can be easily verified. □

Now, we can apply Theorem 2.6.8 to enumerate cubic Cayley pairs and cubic Cayley triples.

Lemma 2.6.9 *The number of non-isomorphic Cayley pairs where exactly one generator is an involution, $I'_{2,n}$, is given by $[xy]Z(\mathbb{S}_n..X_n) + [y]Z(\mathbb{S}_n..X_n)$.*

Proof. $[xy]Z(\mathbb{S}_n..X_n)$ is the number of Cayley pairs where exactly one generator is an involution, not including the identity. $[y]Z(\mathbb{S}_n..X_n)$ is the number of Cayley generators that are involutions. These involutions, when paired with the identity element, are added to $[xy]Z(\mathbb{S}_n..X_n)$ to obtain the result. □

Lemma 2.6.10 *The number of non-isomorphic Cayley triples where all generators are involutions, $I'_{3,n}$, is given by $[y^3]Z(\mathbb{S}_n[X_n])$.*

Proof. By definition $[y^3]Z(\mathbb{S}_n[X_n])$ is the number of Cayley triples where all 3 generators are involutions, not including the identity. Further, notice that if we were allowed to include the identity, the resulting graph would have degree 2. Therefore, we have our result. □

The enumeration results are due to Curtin [9] and are tabulated in Table 2.4.

n	$[xy]Z$	$[y]Z$	$I'_{2,n}$	$I'_{3,n}$
1	0	0	0	0
2	0	0	0	0
3	1	1	2	1
4	7	2	9	10
5	24	2	26	37
6	95	3	98	197
7	267	3	270	676
8	909	4	913	3094
9	2641	4	2645	12022
10	8969	5	8974	55912

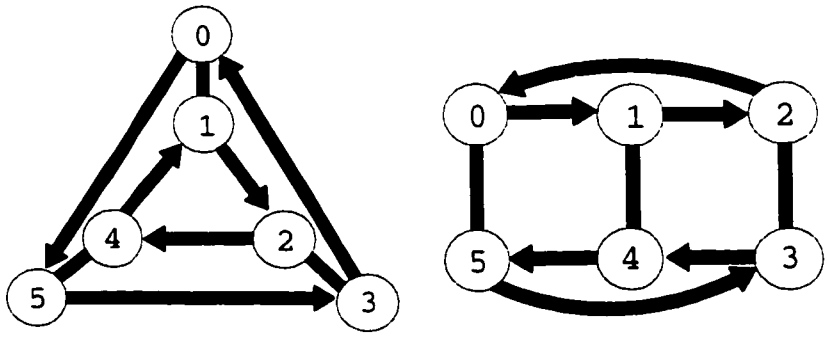
Table 2.4: Enumerations of Non-Isomorphic Cubic Cayley Pairs and Triples From Formula

Notice again that these values are exactly the same as the numbers we found by exhaustive computer search that are documented in Table 2.2.

Chapter 3

Graph Isomorphism

An *isomorphism* from a graph G to a graph H is a one-to-one mapping of the vertices of G onto the vertices of H that preserves adjacency. If there exists an isomorphism from G to H we say G and H are *isomorphic*. In other words, two graphs are isomorphic if the vertices of one can be relabeled to match the vertices of the other in a way that preserves adjacency. The definition of isomorphism can easily be modified for digraphs. For example, the Cayley digraphs drawn in Figure 3.1 are isomorphic.



(a) $\overrightarrow{Cay}(\{(12),(012)\})$ (b) $\overrightarrow{Cay}(\{(234),(01)(34)\})$

Figure 3.1: Isomorphic Cayley Digraphs

Notice that the arc colors are omitted since we are only concerned with adjacency. Also, notice that there may be more than one non-isomorphic Cayley pair that generates a particular Cayley digraph. For example, the Cayley digraph drawn in Figure 3.1(b) is generated by Cayley pair $\{(234),(01)(34)\}$ and by Cayley pair $\{(234),(01)(34)(56)\}$.

Define an *automorphism* of a graph G as an isomorphism from G to G. In

other words, it is a permutation of the vertices of G that preserves adjacency. Further, for Cayley graph G, let $li(G)$ be the minimum value of n, the number of generator symbols, where $\sigma, \tau \in \mathbb{S}_n$ and $\overrightarrow{Cay}(\{\sigma, \tau\})$ is isomorphic to G. This definition can easily be extended to k-generated Cayley graphs for any k and to undirected Cayley graphs.

3.1 Testing for Isomorphism

Testing for isomorphism is important since it will allow us to reduce the number of graphs to test for other properties such as Hamiltonicity. This is due to the fact that if graphs G and H are isomorphic and G is Hamiltonian, then H is Hamiltonian. Nauty [21] is a program written by Brendan McKay that generates automorphism groups of graphs and digraphs. It can also produce canonic labelings. Of greatest relevance to this research is Nauty's ability to check isomorphism between large graphs quickly. The algorithm first canonically labels each graph, then checks for isomorphism. The idea is that once graphs are labeled canonically, it is much easier to test for isomorphism since we only need to check if the adjacency lists are identical. The finer details of this process are beyond the scope of this thesis and thus are omitted.

Consider the digraphs given in Figure 3.1. To test these digraphs for isomorphism using Nauty we would use the input given in Figure 3.2.

Nauty will return the isomorphism if the graphs are isomorphic and "H and H' are Different" if the graphs are non-isomorphic. With the input of Figure 3.2, Nauty will return "0-0 1-5 2-3 3-2 4-4 5-1" which is the isomorphism that maps the second graph to the first. The algorithm is remarkably fast for modest graph sizes (i.e., $|V(G)| < 1000$).

3.2 Data

Pairs of Cayley digraphs generated previously were tested for isomorphism using Nauty as described in the previous section. A huge amount of data

```
c -a -m
n = 6 g
0:  5 1;
1:  2 0;
2:  4 3;
3:  0 2;
4:  1 5;
5:  3 4;
x @
g
0:  5 1;
1:  4 2;
2:  3 0;
3:  2 4;
4:  1 5;
5:  0 3;
x
##
q
```

Figure 3.2: Isomorphism Checking Using Nauty

was generated. Easy to read tables that include the lexicographically smallest Cayley digraph in each equivalence class of isomorphic Cayley digraphs as well as some properties of these digraphs can be found under the heading "Non-Isomorphic 2-generated Cayley Digraphs" at

$$http://www.theory.csc.uvic.ca/\sim cos/cayley/.$$

These tables list all non-isomorphic 2-generated Cayley digraphs G where $n \leq 9$ and $|V(G)| < 1000$. Enumerations are shown in Table 3.1. Only cubic Cayley graphs were generated for $n = 9$.

$\|V(G)\|$	$li(G) = 3$	4	5	6	7	8	9*	total
3	1	0	0	0	0	0	0	1
4	0	2	0	0	0	0	0	2
5	0	0	2	0	0	0	0	2
6	2	0	4	0	0	0	0	6

Table 3.1: Enumerations of Non-Isomorphic 2-Generated Cayley Digraphs
* Only those pairs where exactly one generator is an involution

$\|V(G)\|$	$li(G) = 3$	4	5	6	7	8	9*	total
7		0	0	0	3	0	0	3
8		2	0	2	0	4	0	8
9		0	0	1	0	0	0	1
10		0	2	0	7	0	0	9
12		3	2	0	15	0	0	20
14		0	0	0	2	0	2	4
15		0	0	0	0	12	0	12
16		0	0	0	0	12	0	12
18		0	0	7	0	0	2	9
20		0	6	0	2	0	2	10
21		0	0	0	5	0	0	5
24		5	0	6	14	10	1	36
28			0	0	0	0	1	1
30			0	0	0	18	0	18
32			0	0	0	11	0	11
36			0	5	3	7	0	15
40			0	0	6	0	7	13
42			0	0	13	0	0	13
48			0	5	0	33	0	38
54			0	0	0	0	6	6
56			0	0	0	9	0	9
60			12	0	0	33	1	46
64			0	0	0	10	0	10
72			0	6	25	4	2	37
80			0	0	0	0	2	2
84			0	0	0	0	4	4

Table 3.1: Enumerations of Non-Isomorphic 2-Generated Cayley Digraphs
* Only those pairs where exactly one generator is an involution

$\|V(G)\|$	$li(G) = 3$	4	5	6	7	8	9*	total
96			0	0	0	39	0	39
108			0	0	0	0	3	3
120			19	0	17	5	5	46
144				0	9	4	8	21
162				0	0	0	4	4
168				0	35	56	0	91
180				0	0	43	0	43
192				0	0	49	0	49
216				0	0	0	8	8
240				0	16	0	17	33
288				0	0	50	0	50
324				0	0	0	6	6
336				0	0	68	10	78
360				0	20	99	0	119
384				0	0	18	0	18
432				0	0	0	8	8
480				0	0	0	28	28
504				0	0	0	14	14
576				0	0	127	0	127
648				0	0	0	16	16
720				84	0	140	3	227
total	3	12	47	116	192	861	160	1391

Table 3.1: Enumerations of Non-Isomorphic 2-Generated Cayley Digraphs
* Only those pairs where exactly one generator is an involution

Notice that there are many values of $|V(G)|$ for which Cayley graphs do not occur. By definition, Cayley graphs can only occur if $|V(G)|$ is a divisor

of $n!$. However, this is not to say that if $|V(G)|$ is a divisor of $n!$ that there is always a Cayley graph of that description. For example, when $n = 4$, there are no 2-generated Cayley graphs where $|V(G)| = 6$, even though 6 is a divisor of $4! = 24$.

Unfortunately, as we will show later, we can only prove that this data contains all 2-generated Cayley digraphs G where $|V(G)| \leq 8$. Drawings of all non-isomorphic 2-generated Cayley digraphs G where $|V(G)| \leq 5$ are shown in Figure 3.3.

The same isomorphism analysis was completed on the set of class 1 cubic Cayley graphs where $n \leq 9$. Enumerations are given in Table 3.2.

$\|V(G)\|$	$li(G) = 3$	4	5	6	7	8	9	total
4	0	1	0	0	0	0	0	1
6	1	0	1	0	0	0	0	2
8		1	0	0	0	1	0	2
10		0	1	0	1	0	0	2
12		1	1	0	1	0	0	3
14		0	0	0	1	0	1	2
16		0	0	0	0	3	0	3
18		0	0	2	0	0	1	3
20		0	1	0	1	0	1	3
24		2	0	2	3	0	0	7
28			0	0	0	0	1	1
30			0	0	0	4	0	4
32			0	0	0	2	0	2
36			0	2	0	1	0	3
40			0	0	1	0	4	5
42			0	0	2	0	0	2
48			0	2	0	6	0	8

Table 3.2: Enumerations of Non-Isomorphic Class 1 Cubic Cayley Graphs

$\|V(G)\|$	$li(G) = 3$	4	5	6	7	8	9	total
54			0	0	0	0	4	4
60			3	0	0	3	1	7
64			0	0	0	2	0	2
72			0	2	3	0	1	6
80			0	0	0	0	1	1
84			0	0	0	0	2	2
96			0	0	0	6	0	6
108			0	0	0	0	3	3
120			6	0	8	2	5	21
144				0	2	0	3	5
162				0	0	0	3	3
168				0	5	0	0	5
180				0	0	3	0	3
192				0	0	4	0	4
216				0	0	0	5	5
240				0	4	0	17	21
288				0	0	4	0	4
324				0	0	0	6	6
336				0	0	22	10	32
360				3	0	15	0	18
384				0	0	4	0	4
432				0	0	0	4	4
480				0	0	0	17	17
504				0	0	0	14	14
576				0	0	9	0	9
648				0	0	0	14	14

Table 3.2: Enumerations of Non-Isomorphic Class 1 Cubic Cayley Graphs

$\|V(G)\|$	$li(G) = 3$	4	5	6	7	8	9	total
720				4	0	18	3	25
1080					0	0	3	3
1152					0	14	0	14
1296					0	0	8	8
1344					0	5	0	5
1440					0	3	15	18
1512					0	0	14	14
2160					0	0	11	11
2520					15	0	0	15
2880					0	0	9	9
4320					0	0	6	6
total	1	5	13	17	47	131	187	401

Table 3.2: Enumerations of Non-Isomorphic Class 1 Cubic Cayley Graphs

Similarly, isomorphism analysis was completed on the set of class 2 cubic Cayley graphs where $n \leq 7$. Enumerations are given in Table 3.3.

$\|V(G)\|$	$li(G) = 4$	5	6	7	total
8	0	0	1	0	1
10	0	0	1	0	1
12	0	1	0	1	2
14	0	0	0	2	2
20	0	0	0	2	2
24	1	0	3	5	9
36		0	4	0	4
48		0	8	4	12

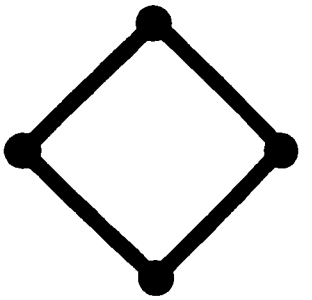
Table 3.3: Enumerations of Non-Isomorphic Class 2 Cubic Cayley Graphs

$\|V(G)\|$	$li(G) = 4$	5	6	7	total
60		4	2	0	6
72		0	4	2	6
120		7	4	22	33
144			0	10	10
168			0	3	3
240			0	26	26
360			7	0	7
720			10	0	10
2520				5	5
total	1	12	44	82	139

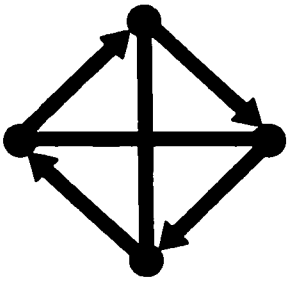
Table 3.3: Enumerations of Non-Isomorphic Class 2 Cubic Cayley Graphs

The results from Tables 3.2 and 3.3 were compared to enumerations of non-isomorphic cubic Cayley graphs computed by Brendan McKay and Gordon Royle [24]. There are two important observations to note. First, it is possible that a graph listed among the non-isomorphic class 1 cubic Cayley graphs is isomorphic to a graph listed among the non-isomorphic class 2 cubic Cayley graphs (since we have only tested graphs from each class for isomorphism). For example, $Cay(\{(01)(23), (0123)\})$ is isomorphic to $Cay(\{(13)(45), (01)(23)(45), (02)(45)\})$ but both are listed among non-isomorphic cubic Cayley graphs for $|V(G)| = 8$ of their respective class. Thus, for $|V(G)| = 8$ there are only two non-isomorphic cubic Cayley graphs even though there are two non-isomorphic class 1 cubic Cayley graphs and one non-isomorphic class 2 cubic Cayley graph. Second, the data computed by McKay and Royle includes all non-isomorphic cubic Cayley graphs where $li(G)$ is unrestricted. However, our data includes only non-isomorphic cubic Cayley graphs

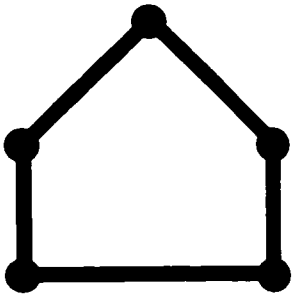
(a) (012),(021)



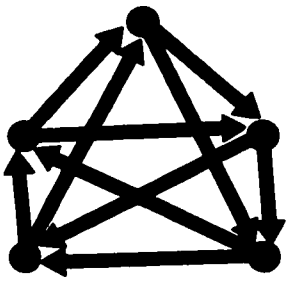
(b) (23),(01)



(c) (01)(23),(0213)



(d) (01234),(04321)



(e) (01234),(02413)

Figure 3.3: Non-Isomorphic 2-Generated Cayley Digraphs

where $li(G) \leq 9$. Therefore, some of McKay and Royle's enumerations are larger than our enumerations for a specific value of $|V(G)|$. For example, for $|V(G)| = 16$, McKay and Royle enumerated four non-isomorphic cubic Cayley graphs, whereas we only generated three non-isomorphic cubic Cayley graphs. The missing graph is $Cay(\{(01234567), (89)\})$. Given these two observations, for $|V(G)| \leq 16$, our data is consistent with the data of McKay and Royle.

3.3 Results

Results that characterize when Cayley graphs are isomorphic are given in Chapter 4 within a context of color isomorphism which is a stronger condition than isomorphism. Some graph theoretic and enumeration results follow.

3.3.1 Graph Theoretic Results

Define the *greatest common divisor* of positive integers r and t, written $gcd(r,t)$, as the largest positive integer x such that $xk_1 = r$ and $xk_2 = t$ where k_1 and k_2 are positive integers. Similarly, the *least common multiple* of positive integers r and t, written $lcm(r,t)$, is the smallest positive integer x such that $rk_1 = tk_2 = x$ where k_1 and k_2 are positive integers.

First, we establish that all 2-generated Cayley digraphs are diregular.

Lemma 3.3.1 *All 2-generated Cayley digraphs are (2,2)-diregular.*

Proof. Our algorithm for constructing adjacency lists creates one outgoing arc from each vertex for each generator. Similarly, there is one incoming arc to each vertex for each generator. This can be seen by considering generators σ^{-1} and τ^{-1}, and noticing that outgoing arcs become incoming arcs, and vise versa. □

We can easily extend this result to general Cayley digraphs.

Theorem 3.3.2 *All k-generated Cayley digraphs are (k,k)-diregular.*

Proof. The proof of Lemma 3.3.1 can be easily generalized to k-generated Cayley digraphs. □

Next, we will construct an upper bound for the number of vertices a Cayley digraph on n symbols can have.

Theorem 3.3.3 *For all Cayley digraphs G, $li(G) \leq |V(G)|$.*

Proof. We know that the order of a generator is equal to the least common multiple of the cycle lengths of the generator. Also, the fewest number of vertices a Cayley digraph G can have is the least common multiple of the orders of the generators, call this $x(G)$. We have now established that $x(G) \leq |V(G)|$. It remains to show that $li(G) \leq x(G)$. Suppose $li(G) > x(G)$. This is only possible if part of the cycle structure of the generators is repeated. For example, $\overrightarrow{Cay}(\{(01)(23)(45)(67), (0123)(4567)\})$ is isomorphic to $\overrightarrow{Cay}(\{(01)(23), (0123)\})$. However, this is a contradiction, so we are done. □

This result is interesting because now we can be sure that we have produced all 2-generated Cayley digraphs with fewer than 9 vertices. Theorem 3.3.3 is tight since for instance Cayley pairs that generate the cycle graph, C_n, have n-vertices. For example, $\overrightarrow{Cay}(\{(012), (021)\})$ has 3 vertices. An interesting extension to this question is what is this maximum if these pairs of disjoint n-cycles are not considered? As it turns out, this omission does not better the result. For example, some circulant graphs also have directed n-cycles using n-generator symbols. Therefore, we must consider all generators with up to n symbols to generate all Cayley digraphs with at most n vertices. Therefore, we know that our data is complete for $|V(G)| \leq 8$ since we have produced all 2-generated Cayley digraphs on $n \leq 8$.

3.3.2 Vertex Transitivity Results

A graph is said to be *vertex transitive* if for every pair of vertices, u and v, there is an automorphism that maps u to v. Intuitively, a vertex transitive

graph "looks the same" from every vertex. The following lemma is obvious from the definition of Cayley digraphs [13].

Lemma 3.3.4 *All Cayley digraphs are vertex transitive.*

Notice, however, that the converse is not necessarily true.

Lemma 3.3.5 *Not all vertex transitive graphs are Cayley graphs.*

Proof. It is well known that the Petersen graph is vertex transitive [39]. Further, we know that the Petersen graph is non-Hamiltonian [39]. From our data (see Chapter 5), we know that all cubic Cayley graphs with 10 vertices are Hamiltonian. Therefore, the Petersen graph cannot be a Cayley graph even though it is vertex transitive. □

3.3.3 Enumeration of Non-Isomorphic Cayley Digraphs

The purpose of this section is to gain an understanding of relatively how many graphs are Cayley graphs. To this end, we will present Pólya theory (generalization of Burnside's Lemma) formulae for enumerating various collections of graphs and digraphs.

As an introduction, the equation for the number of unlabeled digraphs [14] can be used to provide an upper bound for the number of non-isomorphic Cayley digraphs. This well known enumeration is stated below. We will use p to denote the number of vertices instead of $|V(G)|$ to reduce the number of subscripts in our formulae.

Theorem 3.3.6 *Let $d_p(x)$ be the generating function for the enumeration of unlabeled digraphs with p vertices, then,*

$$d_p(x) = Z(\mathbb{S}_p^{[2]}, 1 + x)$$

where,

$$Z(\mathbb{S}_p^{[2]}) = \frac{1}{p!} \sum_{j_1+2j_2+\cdots+pj_p=p} \frac{p!}{\prod k^{j_k} j_k!} \prod_{k=1}^{p} s_k^{(k-1)j_k+kj_k(j_k-1)} \prod_{r<t} s_{lcm(r,t)}^{2j_r j_t gcd(r,t)}$$

From this, we can easily determine the number of unlabeled digraphs with p vertices, d_p, by summing the coefficients in $d_p(x)$. Enumerations for small values of p are given in Table 3.4.

This enumeration leads to an upper bound on the number of non-isomorphic Cayley digraphs with a given number of vertices.

Lemma 3.3.7 *The number of non-isomorphic Cayley digraphs with p vertices is no more than* d_p.

Proof. The number of non-isomorphic Cayley digraphs with p vertices is obviously a subset of the number of unlabeled digraphs with p vertices. □

This result can be improved by counting only the connected digraphs, c_p, since all Cayley digraphs are connected by definition. This is done using a method of Harary and Palmer [14] for counting connected graphs and connected digraphs.

Theorem 3.3.8 *The generating functions* $d_p(x)$ *and* $c_p(x)$ *for graphs and connected digraphs satisfy*

$$1 + d_p(x) = exp \sum_{k=1}^{\infty} c_p(x^k)/k.$$

Note that c_p is the sum of all coefficients in $c_p(x)$. Enumerations are given in Table 3.4 and an improved result follows.

Lemma 3.3.9 *The number of non-isomorphic Cayley digraphs with p vertices is no more than* c_p.

Proof. All Cayley digraphs are connected because by definition a Cayley digraph is a connected component of the digraph generated by Algorithm 2.3. The number of non-isomorphic Cayley digraphs with p vertices is obviously a subset of the number of connected graphs with p vertices. □

Note that this is an enumeration of all non-isomorphic Cayley digraphs, not just the 2-generated or 3-generated ones. This bound could be tightened if we enumerate only vertex transitive digraphs. A paper of McKay and Royle [23] gives insight into how to count all vertex transitive digraphs. However, no formula is given. Further, a result of McKay and Praeger [22] gives results pertaining to the number of vertex transitive digraphs that are not Cayley digraphs. A conjecture given in this work is useful.

Conjecture 3.3.10 *The great majority of vertex transitive digraphs are Cayley digraphs.*

If this conjecture were true, we would know that the number of vertex transitive digraphs gives a good upper bound to the number of Cayley digraphs. However, the problem of counting vertex transitive digraphs is currently unsolved.

3.3.4 Enumeration of Non-Isomorphic 2-Generated Cayley Digraphs

The bound of Lemma 3.3.9 gives no insight into how many non-isomorphic 2-generated Cayley digraphs there are except to say they are a subset of all non-isomorphic Cayley digraphs. A result of Ramanath and Walsh [29] can be extended and applied to this problem. Integers r and t are *relatively prime* if $gcd(r,t) = 1$. Define the *Euler totient function*, $\varphi(n)$, as the number of positive integers less than n which are relatively prime to n.

Theorem 3.3.11 *Let $d_p^{2,2}(x)$ be the generating function for the enumeration*

of unlabeled (2,2)-diregular digraphs with p vertices, then,

$$d_p^{2,2}(x) = \prod_{u=1}^{t} Z(\mathbb{S}_{k_i}) Z(D_{n_i})$$

where

$$Z(D_n) = \frac{1}{2n}\left(R + \sum_{r|n} \varphi\left(\frac{n}{r}\right) y_{n/r}^r z_{n/r}^r\right)$$

$$R = \begin{cases} n y_1 z_1 y_2^{(n-1)/2} z_2^{(n-1)/2} & \text{if } n \text{ is odd} \\ \frac{n}{2}(y_1^2 z_2 + z_1^2 y_2) y_2^{(n-2)/2} z_2^{(n-2)/2} & \text{otherwise} \end{cases}$$

$$Z(\mathbb{S}_k) = \frac{1}{k!} \sum_{j_1+2j_2+\cdots+kj_k=k} \frac{k!}{1^{j_1} 2^{j_2} \cdots k^{j_k} j_1! j_2! \cdots j_k!} x_1^{j_1} x_2^{j_2} \cdots x_k^{j_k}.$$

The number of unlabeled (2,2)-diregular digraphs with p vertices, $d_p^{2,2}$, is the sum of the coefficients of $d_p^{2,2}(x)$. Enumerations are given in Table 3.4 and an upper bound results.

Lemma 3.3.12 *The number of non-isomorphic 2-generated Cayley digraphs with p vertices is no more than* $d_p^{2,2}$.

Proof. The number of non-isomorphic 2-generated Cayley digraphs with p vertices is obviously a subset of the number of unlabeled (2,2)-diregular digraphs with p vertices. □

This result can be improved by only counting the connected (2,2)-diregular digraphs, $c_p^{2,2}$. We apply Theorem 3.3.8 to obtain the following result.

Lemma 3.3.13 *The number of non-isomorphic 2-generated Cayley digraphs with p vertices is no more than* $c_p^{2,2}$

Proof. Non-isomorphic 2-generated Cayley digraphs are obviously a subset of the set of all unlabeled (2,2)-diregular connected digraphs. □

Table 3.4 contains enumerations based on these formulae.

p	d_p	c_p	$d_p^{2,2}$	$c_p^{2,2}$
1	1	1	0	0
2	3	2	1	1
3	16	13	3	3
4	218	199	8	7
5	9608	9364	27	24
6	1540944	1530843	131	117
7	882033440	880471142	711	663
8	1793359192848	1792473955306	5055	4824

Table 3.4: Enumerations of Digraphs

Note that this result does not extend to general (k,k)-diregular digraphs. In fact, the result is dependent on the special structure of (2,2)-diregular digraphs. Further, the problem of enumerating vertex transitive (2,2)-diregular digraphs is currently unsolved.

3.3.5 Enumeration of Non-Isomorphic Cubic Cayley Graphs

We now consider enumerating non-isomorphic cubic Cayley graphs using a result of Parthasarathy [14]. Define θ(W(f)) as the monomial that results from reordering of the exponents in W(f) in non-increasing order while stating the variables in increasing order.

Theorem 3.3.14 *The generating function that enumerates graphs with degree sequence* $x_1, x_2, \ldots, x_p$, $N(x_1, x_2, \ldots, x_p)$, *is*

$$N(x_1, x_2, \ldots, x_p) = \frac{1}{p!} \sum_{y \epsilon E_2^{S_p^{(2)}}} \theta\left(\sum_{f=yf} W(f)\right) \tag{3.1}$$

where

$$\sum_{f=yf} W(f) = \prod_{z_r,z_s} \left(1 + \prod_{i\epsilon z_r} x_i^{s/(r,s)} \prod_{j\epsilon z_s} x_j^{r/(r,s)}\right)^{(r,s)}$$
$$\prod_{z_r,r \text{ even}} \left(1 + \prod_{i\epsilon z_r} x_i\right)\left(1 + \prod_{i\epsilon z_r} x_i^2\right)^{(r-2)/2}$$
$$\prod_{z_r,r \text{ odd}} \left(1 + \prod_{i\epsilon z_r} x_i^2\right)^{(r-1)/2}$$

We can apply this result to count cubic graphs, and give an upper bound on the number of cubic Cayley graphs.

Lemma 3.3.15 *The number of non-isomorphic cubic graphs with p vertices is given by* $N(\underbrace{3,3,\ldots,3}_{p})$. *This gives an upper bound of the number of non-isomorphic cubic Cayley graphs.*

Proof. Trivial by the formula above. This is obviously an upper bound since the set of all non-isomorphic cubic Cayley graphs is a subset of the set of all non-isomorphic cubic graphs. □

It is evident that counting non-isomorphic Cayley graphs, even in restricted cases, is difficult. Perhaps the best chance for a useful formula would be to attempt to count the number of vertex transitive graphs, or a restriction thereof.

Chapter 4

Graph Color Isomorphism

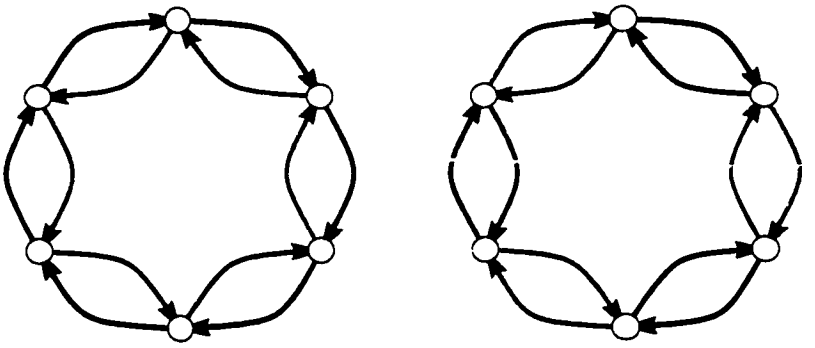
In this chapter, we will consider 2-generated Cayley digraphs in which arcs are colored based upon which of the two generators created them. The resulting graph is a 2-colored (2,2)-diregular digraph.

A *color isomorphism* from a colored graph G to a colored graph H is a one-to-one mapping of the vertices of G onto the vertices of H that preserves adjacency and color. Note that this is the same as the definition of isomorphism except that corresponding edges from G and H must be of the same color (i.e., they must have been generated from the same generator). If there exists a color isomorphism from G to H we say G and H are color isomorphic. More simply, two colored graphs are color isomorphic if the vertices of one can be relabeled to match the vertices of the other in a way that preserves adjacency and color. This definition can be easily extended to colored digraphs. For example, the Cayley digraphs drawn in Figure 4.1 are not color isomorphic since colors cannot be preserved. Note, however, that these Cayley digraphs are isomorphic.

As a second example, consider the Cayley digraphs of Figure 4.2. These Cayley digraphs are not isomorphic, and therefore, they are not color isomorphic.

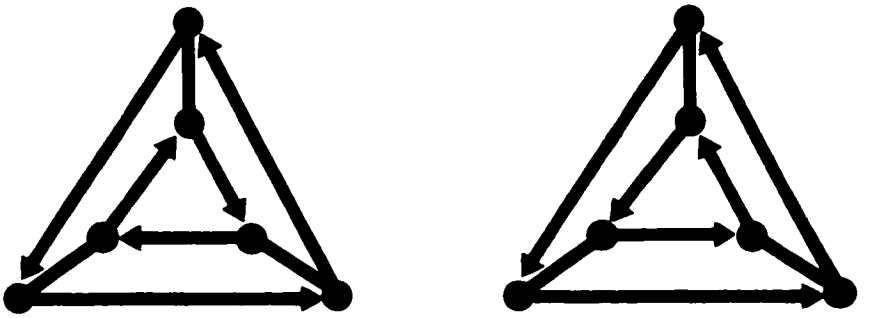
For Cayley graph G, let $lci(G)$ be the minimum value of n, the number of generator symbols, where $\sigma, \tau \in \mathbb{S}_n$ and $\overrightarrow{Cay}(\{\sigma, \tau\})$ is color isomorphic to G. This definition can easily be extended to k-generated Cayley digraphs for any k and to undirected Cayley graphs.

Define the *order* of generator σ, written $order(\sigma)$, to be the smallest $n > 1$ such that $\sigma^n = \sigma$ (note that $order(I) = 1$). This is the least common multiple



(a) $\overrightarrow{Cay}(\{(012345),(054321)\})$ (b) $\overrightarrow{Cay}(\{(23)(45),(02)(14))\})$

Figure 4.1: Non-Color Isomorphic Cayley Digraphs



(a)
$\overrightarrow{Cay}(\{(23)(45),(023)(145)\})$

(b)
$\overrightarrow{Cay}(\{(01)(23)(45),((024)(135)\})$

Figure 4.2: Non-Color Isomorphic Cayley Digraphs

of all the cycle lengths in σ (recall that all generators are permutations).

In a *depth first search* (DFS) we explore vertices adjacent to the most recently discovered vertex that has unexplored edges [39]. That is, extremes are searched first. A *depth first search numbering* results from labeling vertices in the order they are visited in a depth first search. In this research, we will always start the depth first search at the identity permutation.

4.1 Testing for Color Isomorphism

Although we have tested Cayley digraphs for isomorphism, it is also useful to check for color isomorphism since Cayley digraphs are inherently colored. An algorithm for determining color isomorphism between Cayley digraphs follows.

4.1.1 Algorithm

A fast algorithm for determining whether or not two Cayley digraphs are color isomorphic is needed. In this case, a canonic labeling is achieved by relabeling based on a depth first search numbering. Here we will assume that the i^{th} column of the adjacency list corresponds to the arcs created by the i^{th} generator, for $1 \leq i \leq |X|$. Now, the adjacency lists of the resulting canonic digraphs are identical (possibly with the columns of the adjacency lists swapped) if and only if they are color isomorphic. Algorithm 4.1 tests for color isomorphism.

```
Given Cayley digraphs G and H
begin
    Relabel the vertices of G using a DFS numbering
    Relabel the vertices of H using a DFS numbering
    H' = H with adjacency columns reversed
    if G = H or G = H' then
        return YES;
    else
        return NO;
    endif
end
```

Algorithm 4.1: Determining Color Isomorphism of Cayley Digraphs

4.1.2 Example

Consider the digraphs of Figure 4.1. We relabel the vertices using a depth first search numbering starting at any vertex, say the top one in the picture. Both graphs turn out to be labeled in a clockwise fashion starting at the top vertex. Obviously, $G \neq H$ and $G \neq H'$ so the digraphs are not color isomorphic.

4.1.3 Correctness

A proof of correctness of the color isomorphism algorithm is simple given the above discussion.

Theorem 4.1.1 *Given 2-generated Cayley digraphs G and H, Algorithm 4.1 determines whether or not G and H are color isomorphic in linear time.*

Proof. It is obvious that relabeling based upon a depth first search numbering identifies the canonic representative of a Cayley digraph. This is due to the fact that a depth first search of two color isomorphic Cayley digraphs, starting at the identity, will result in the same unlabeled spanning tree. Now, G and H (after relabeling) belong to the same color isomorphism equivalence class if and only if the adjacency lists of the relabeled graphs are identical (possibly with the columns reversed). Further, since the depth first search algorithm for graphs with fixed degree runs in linear time, and testing to see if two graphs are identical takes linear time, Algorithm 4.1 runs in linear time. □

This algorithm can be easily extended to test for color isomorphism between Cayley digraphs generated by more than two generators. After relabeling the digraphs based upon a DFS numbering, the Cayley digraphs are color isomorphic if and only if the adjacency lists are identical (possibly with the columns permuted). This modified algorithm runs in linear time for a fixed number of generators.

4.2 Data

Pairs of Cayley digraphs generated previously were tested for color isomorphism using Algorithm 4.1. A huge amount of data was generated. Easy to read tables that include the lexicographically smallest Cayley digraph in each equivalence class of color isomorphic graphs as well as some properties of these graphs can be found under the heading "Non-Color Isomorphic 2-generated Cayley Digraphs" at

$$http://www.theory.csc.uvic.ca/\sim cos/cayley/.$$

These tables list all non-color isomorphic 2-generated Cayley digraphs G where $n \leq 9$ (only cubic Cayley graphs for $n = 9$) and $|V(G)| < 1000$. Enumerations are shown in Table 4.1.

$\|V(G)\|$	$lci(G) = 3$	4	5	6	7	8	9	total
3	1	0	0	0	0	0	0	1
4	0	3	0	0	0	0	0	3
5	0	0	2	0	0	0	0	2
6	2	0	5	0	0	0	0	7
7		0	0	0	3	0	0	3
8		2	0	2	0	7	0	11
9		0	0	1	0	0	0	1
10		0	2	0	8	0	0	10
12		3	2	0	19	0	0	24
14		0	0	0	2	0	2	4
15		0	0	0	0	12	0	12
16		0	0	0	0	14	0	14
18		0	0	7	0	2	1	10
20		0	7	0	2	0	2	11
21		0	0	0	5	0	0	5

Table 4.1: Enumerations of Non-Color Isomorphic 2-Generated Cayley Digraphs

$\|V(G)\|$	$lci(G) = 3$	4	5	6	7	8	9	total
24		5	0	7	15	10	1	38
28			0	0	0	0	1	1
30			0	0	0	19	0	19
32			0	0	0	12	0	12
36			0	6	3	7	0	16
40			0	0	7	0	7	14
42			0	0	13	0	0	13
48			0	5	0	37	0	42
54			0	0	0	0	6	6
56			0	0	0	9	0	9
60			12	0	0	36	1	49
64			0	0	0	10	0	10
72			0	6	26	4	2	38
80			0	0	0	0	2	2
84			0	0	0	0	4	4
96			0	0	0	39	0	39
108			0	0	0	0	3	3
120			31	0	31	12	6	80
144				0	9	4	8	21
162				0	0	0	4	4
168				0	35	56	0	91
180				0	0	43	0	43
192				0	0	51	0	51
216				0	0	0	8	8
240				0	31	0	19	50
288				0	0	51	0	51

Table 4.1: Enumerations of Non-Color Isomorphic 2-Generated Cayley Digraphs

$\|V(G)\|$	$lci(G) = 3$	4	5	6	7	8	9	total
324				0	0	0	6	6
336				0	0	111	15	126
360				32	0	181	0	213
384				0	0	27	0	27
432				0	0	0	8	8
480				0	0	0	28	28
504				0	0	0	14	14
576				0	0	139	0	139
648				0	0	0	16	16
720				85	0	142	3	230
total	3	13	61	151	209	1035	167	1639

Table 4.1: Enumerations of Non-Color Isomorphic 2-Generated Cayley Digraphs

4.3 Results

Using our tables of non-color isomorphic Cayley graphs as a basis, we were able to develop many interesting results. They are documented in this section.

4.3.1 Basic Results

First, we link graph color isomorphism with graph isomorphism.

Lemma 4.3.1 *If two Cayley graphs are color isomorphic, then they are isomorphic.*

Proof. If we ignore colors then any pair of color isomorphic graphs are also isomorphic. □

Corollary 4.3.2 *There are at least as many non-color isomorphic Cayley*

graphs as there are non-isomorphic Cayley graphs.

Proof. Obvious by Lemma 4.3.1. □

Notice that due to Lemma 4.3.1 many results pertaining to color isomorphism also apply to isomorphism.

Next, we explore results based upon the order of the generators and the number of vertices in the resulting graphs.

Lemma 4.3.3 *$\overrightarrow{Cay}(\{\sigma,\tau\})$ is not color isomorphic to $\overrightarrow{Cay}(\{\sigma',\tau'\})$ if $\{order(\sigma), order(\tau)\} \neq \{order(\sigma'), order(\tau')\}$.*

Proof. Without loss of generality, assume $order(\sigma) < order(\sigma') \leq order(\tau')$. Therefore, the first graph contains a σ-colored cycle of length $order(\sigma)$. However, the size of the smallest cycle colored by a single color that the second graph can contain is $order(\sigma')$. Since $order(\sigma) < order(\sigma')$, these graphs cannot be color isomorphic. □

For example, $\overrightarrow{Cay}(\{(0123)(4567), (0321)(4765)\})$ is not color isomorphic to $\overrightarrow{Cay}(\{(01)(23)(45)(67), (0213)(4657)\})$ since $\{4,4\} \neq \{2,4\}$.

Lemma 4.3.4 *The Cayley digraph G generated by $\{\sigma,\tau\}$ is not (color) isomorphic to the Cayley digraph H generated by $\{\sigma',\tau'\}$ if $|V(G)| \neq |V(H)|$*

Proof. Graphs with a different number of vertices cannot be color isomorphic, nor can they be isomorphic. □

For example, the Cayley digraph G generated by {(23),(01)} is not (color) isomorphic to the Cayley digraph H generated by {(01)(23)(45),(02)(14)(35)} since $|V(G)| = 4 \neq 6 = |V(H)|$.

Finally, we explore results based on the structure of the generators themselves.

Lemma 4.3.5 *$\overrightarrow{Cay}(\{\sigma,\tau\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{\sigma',\tau'\})$ if Cayley pair (σ,τ) is isomorphic to Cayley pair (σ',τ').*

Proof. Since the Cayley pairs are isomorphic, the graphs will be color isomorphic and therefore also isomorphic. This is because it does not matter how we label the generator symbols, as long as the labels are consistent between the generators. □

For example, $\overrightarrow{Cay}(\{(012),(021)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(120),(120)\})$ on relabeling $0 \mapsto 1$, $1 \mapsto 0$ and $2 \mapsto 2$.

Lemma 4.3.6 *$\overrightarrow{Cay}(\{\sigma,\tau\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{\sigma',\tau'\})$ if $\sigma' = \sigma(n)$ and $\tau' = \tau(n)$.*

In a slight abuse of notation, note that $\sigma(n)$ means append a 1-cycle onto σ.

Proof. Adding 1-cycles will not change the graph structure. □

For example, $\overrightarrow{Cay}(\{(012),(021)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(012)(3),(021)(3)\})$ since we simply added a 1-cycle to each generator.

Notice that Lemma 4.3.5 and Lemma 4.3.6 can be combined so that if we add a 1-cycle on any symbol and update the other symbols appropriately we will have a Cayley graph that is (color) isomorphic to the original.

Lemma 4.3.7 *$\overrightarrow{Cay}(\{\sigma,\tau\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{\sigma\sigma',\tau\})$ where*

$$\sigma = (a_1 a_2 \cdots)(b_1 b_2 \cdots)\cdots$$
$$\sigma' = (a_1+n, a_2+n, \ldots)(b_1+n, b_2+n, \ldots)\cdots.$$

Proof. Since σ and σ' have the same structure, they will create the same graph. Therefore, concatenating these together will not change the graph structure. □

For example, $\overrightarrow{Cay}(\{(012),(021)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(012)(345),(021)\})$.

4.3.2 Involution Results

Recall that a non-identity generator σ is an involution if $order(\sigma) = 2$. Define a $2-$*color alternating cycle graph* as a graph induced by a cycle where edges are colored with one of two colors such that incident edges have different colors. For example, the Cayley graph of Figure 4.1(b) is a 2-color alternating cycle graph with six vertices. Notice that all such cycles must be of even length. Results concerning Cayley graphs where the generators are involutions follow.

Lemma 4.3.8 *The Cayley digraph G generated by $\{\sigma, \tau\}$ is (color) isomorphic to the Cayley digraph H generated by $\{\sigma', \tau'\}$ if:*

- $|V(G)| = |V(H)|$
- $order(\sigma) = order(\tau) = order(\sigma') = order(\tau') = 2$
- $\sigma \neq \tau$ *and* $\sigma' \neq \tau'$.

Proof. Since involutions have the property $x\sigma^2 = x$, we have a 2-regular undirected graph. Because Cayley graphs are connected, and we ensure $\sigma \neq \tau$, all such graphs will be 2-color alternating cycle graphs with $|V(G)| = |V(H)|$ vertices. Because of this, G and H are color isomorphic and are therefore also isomorphic. □

For example, $\overrightarrow{Cay}(\{(01)(23)(45), (02)(14)(35)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(23)(45), (02)(14)\})$ since both are 2-color alternating cycle graphs with six vertices.

Lemma 4.3.9 $\overrightarrow{Cay}(\{\sigma, \tau\})$ *is (color) isomorphic to* $\overrightarrow{Cay}(\{\sigma', \tau'\})$ *where* $\sigma' = \sigma(n, n+1)$ *and* $\tau' = \tau(n, n+1)$ *if* $order(\sigma) = order(\tau) = 2$.

Proof. As in the proof of Lemma 4.3.8, we know that $\overrightarrow{Cay}(\{\sigma, \tau\})$ is a 2-color alternating cycle graph. If we add a 2-cycle on new symbols to each generator, these symbols will flip at each step, resulting in a 2-color alternating cycle graph. □

For example, $\overrightarrow{Cay}(\{(0)(1)(23),(01)(2)(3)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(0)(1)(23)(45),(01)(2)(3)(45)\})$ since both graphs are 2-color alternating cycle graphs on four vertices.

For these results to be useful in any application, we need to have a way to quickly compute $|V(G)|$. When G is a 2-generated Cayley graph where both generators are involutions there is a very simple way to do this. Starting from the identity, we apply the generators alternatively until the identity results. In other words, $I = I(\sigma\tau)^{n/2}$. The process is shown in Algorithm 4.2

```
Given {σ, τ} where order(σ) = order(τ) = 2
begin
    n = 0; v = I;
    repeat
        v = σ(τv);
        n = n + 2;
    until v = I;
    return n;
end
```

Algorithm 4.2: Computing $|V(G)|$ of Cayley Graph G Generated by $\{\sigma,\tau\}$ Where $order(\sigma) = order(\tau) = 2$

This simple algorithm runs in linear time, $\mathcal{O}(|V(G)|)$.

4.3.3 Higher Order Results

A more complicated result can be derived by generalizing previous results.

Lemma 4.3.10 *$\overrightarrow{Cay}(\{\alpha\beta,\gamma\delta\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{\alpha\beta\beta',\gamma\delta\delta'\})$ if:*

- *β' disjoint from $\alpha\beta$, δ' disjoint from $\gamma\delta$*
- *β and β' have same cycle structure, δ and δ' have same cycle structure*
- *β and δ are permutations of the same elements.*

Proof. Since β and δ are permutations of the same elements and β has the same cycle structure as β' (similarly for δ) the new portion of the generators, β' and δ', simply mimic the portion of the graph that β and δ created. Therefore, the graphs will be identical and therefore are (color) isomorphic. □

For example, $\overrightarrow{Cay}(\{(01)(234), (01)(243)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(01)(234)(567), (01)(243)(576)\})$ since $\beta = (234)$, $\beta' = (567)$, $\delta = (243)$ and $\delta' = (576)$.

Notice that this result is a generalization of Lemma 4.3.7. Also, the restriction that the two Cayley graphs have the same number of vertices (as with involutions) is automatically implied by the construction of the second graph from the first. Therefore, we do not need to calculate $|V(\overrightarrow{Cay}(\{\alpha\beta, \gamma\delta\})|$ in each Cayley graph as we did for graphs where all generators are involutions.

4.3.4 Transitivity of Isomorphism and Color Isomorphism

By definition, all equivalence classes are *transitive*. For example, isomorphism is transitive since G_1 is isomorphic to G_3 whenever G_1 is isomorphic to G_2 and G_2 is isomorphic to G_3. Therefore, we can repeatedly apply the results of this chapter to prove that a graph G is (color) isomorphic to a graph H.

For example, $\overrightarrow{Cay}(\{(012), (01)\})$ is (color) isomorphic to $\overrightarrow{Cay}(\{(021), (01)\})$ by first applying Lemma 4.3.6 then Lemma 4.3.5.

4.3.5 Isomorphism Classes

At this point, we have looked at three types of isomorphism: isomorphic Cayley pairs, isomorphic Cayley graphs and color isomorphic Cayley graphs. There is a nice correspondence that pulls together these three concepts. Notice that if two Cayley pairs are isomorphic, then their associated graphs are also color isomorphic and isomorphic. Also, by Lemma 4.3.1 we know that if two Cayley graphs are color isomorphic then they are isomorphic. These two facts

alone allow us to picture the relationship between these properties much like complexity classes. This relationship is shown in Figure 4.3.

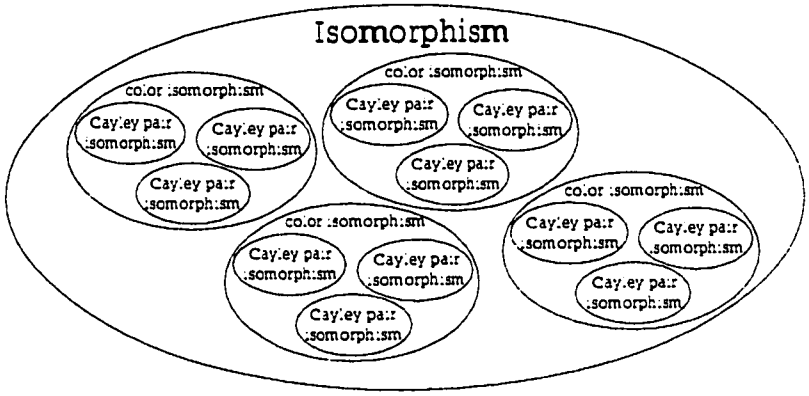


Figure 4.3: Relationship Between Isomorphism Classes

Now, if we wanted to test for the often tough property of graph isomorphism we could first test Cayley pair isomorphism and color isomorphism. If we receive a positive answer the graphs are isomorphic, otherwise, we must test for isomorphism.

4.3.6 Enumeration Results

Counting non-color isomorphic Cayley digraphs is a difficult problem, maybe more difficult than counting non-isomorphic Cayley digraphs. Only trivial bounds are known.

Define $C^{i}_{2,k}$ as the number of non-isomorphic 2-generated Cayley digraphs with k vertices. Similarly, define $C^{ci}_{2,k}$ as the number of non-color isomorphic 2-generated Cayley digraphs with k vertices.

Each non-isomorphic Cayley digraph can be colored in many ways. At each vertex, we must choose the arc to be colored with the first generator, and the arc to be colored with the second generator. This can be done in 2^k ways. However, every coloring obtained in this way is by no means going to translate into a Cayley digraph. Therefore, this is only an upper bound on the number

of non-color isomorphic Cayley digraphs. If we choose the color for each edge to be colored we trivially have the following inequality.

$$C_{2,k}^{ci} \leq 2^k \cdot C_{2,k}^{i} \tag{4.1}$$

Unfortunately, at this time, tighter results are not known.

Chapter 5

Hamiltonicity

A (directed) *Hamilton* cycle is a cycle that visits every vertex in the (directed) graph. We compiled a list of all class 1 cubic Cayley graphs for $n \leq 9$ from our lists of 2-generated Cayley graphs by testing whether or not exactly one of the generators was an involution. Further, we generated all class 2 cubic Cayley graphs for $n \leq 7$ from scratch by generating all non-isomorphic Cayley triples where all three generators are involutions.

All 2-generated Cayley digraphs where $n \leq 8$ were tested for directed Hamiltonicity. Also, all class 1 cubic Cayley graphs where $n \leq 9$ and all class 2 cubic Cayley graphs where $n \leq 7$ were tested for undirected Hamiltonicity.

5.1 Motivation

The Hamiltonicity of Cayley graphs is an interesting problem because although these graphs are easy to describe, their Hamiltonicity in general is still a mystery. We initially began researching the Hamiltonicity of Cayley graphs because of a long standing famous conjecture [40].

Conjecture 5.1.1 *(T.D. Parsons and others) All undirected connected Cayley graphs are Hamiltonian.*

We set out on a journey to find a counterexample to this conjecture. We decided to search through cubic Cayley graphs first since fast algorithms for finding Hamilton cycles in cubic graphs are already known. We did not find a counterexample, however, the results gathered along the way were interesting just the same!

A similar search called the Foster Census [6] was conducted on symmetric cubic graphs. They found only five non-Hamiltonian symmetric cubic graphs G where $|V(G)| \leq 1000$. They are the complete graph on two vertices, the Petersen graph, the Coxeter graph and two graphs derived from the Petersen and Coxeter graphs by replacing each vertex with a triangle. We verified through computer search that none of the above graphs are Cayley graphs. They also conjecture that all symmetric graphs except those listed above are Hamiltonian.

From the Foster Census, we know that all cubic Cayley graphs with fewer than one thousand vertices are Hamiltonian. Therefore, we set out first to find Hamilton cycles for as many cubic Cayley graphs G, where $|V(G)| > 1000$, as possible.

5.2 Complexity of Hamiltonicity Testing

It is well known that the problem of finding a Hamilton cycle in a general graph or digraph is NP-complete [12]. More simply, this means that it is unlikely that there exists a polynomial time algorithm that can solve this problem in all cases. Further, it is known that finding a Hamilton cycle in a cubic graph is also NP-complete [12]. Does this mean that we have no hope? Absolutely not. In fact, not much is known about the complexity of finding Hamilton cycles in cubic Cayley graphs. It turns out that cubic Cayley graphs have nice properties that allow us to eliminate some possible cycles and thus cut down computation time. As an example, consider the partial Cayley graph of Figure 5.1. If edges (U, V) and (V, W) are in a Hamilton cycle, then edge (V, Y) cannot be in the cycle. Further, this forces edges (X, Y) and (Y, Z) to be in the Hamilton cycle. Edges that are in the cycle are drawn blue and the edge that cannot be in the cycle is drawn red.

Observations such as this one lead to remarkably quick backtracking algorithms for finding Hamilton cycles in cubic graphs most of the time. Similar

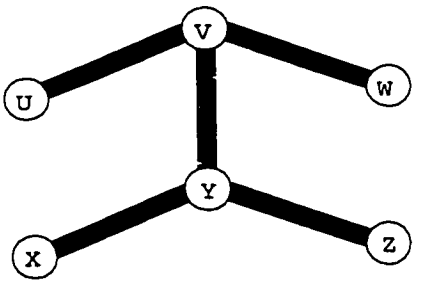


Figure 5.1: Hamiltonicity of Cubic Graphs

tricks are used in algorithms for directed Hamiltonicity.

5.3 Directed Hamiltonicity

Notice that Conjecture 5.1.1 does not apply to directed Cayley graphs. There exist many non-Hamiltonian directed Cayley graphs. To this end, we were interested in determining directed Hamiltonicity for the Cayley digraphs generated earlier.

Many algorithms have been developed for finding Hamilton cycles in directed graphs [38]. A brief survey of these algorithms follows.

DHC by Algluin and Valiant builds up a cycle by adding arcs. If the algorithm is forced to stop before a Hamilton cycle is found, it does a directional rotational transformation. This process continues until a Hamilton cycle is found, or all possibilities have been searched.

DB2A by Brunacci converts a digraph into an undirected graph as in the proof of NP-completeness. Then, we use an undirected Hamiltonian cycle algorithm to find a cycle in this undirected graph. Finally, we convert the cycle back to the form of the original directed graph.

595*Ham* by Martello is a backtracking algorithm that uses several basic pruning techniques to reduce its running time.

In this research, we used a theorem of Rankin (see Section 5.3.1) to tell us that some of the Cayley digraphs were non-Hamiltonian. Then, we used DigHam (see Section 5.3.2) by Brendan McKay to determine the Hamiltonicity

of the Cayley digraphs where Rankin's theorem does not apply.

5.3.1 Rankin's Theorem

2-generated Cayley digraphs were tested to determine their directed Hamiltonicity. A theorem of Rankin [30] was used to characterize many of these digraphs as non-Hamiltonian. The *index* of generator σ, $index(\sigma)$, is the size of the group generated by σ.

Theorem 5.3.1 *$\overrightarrow{Cay}(\{\sigma,\tau\} : G)$ is non-Hamiltonian if $order(\sigma^{-1}\tau)$ is odd and $index(\sigma)$ or $index(\tau)$ is even.*

We give a proof of this theorem, similar to that of Swan [36], but in the language of Cayley graphs.

Proof. (of Theorem 5.3.1)
We will prove a more general theorem, and link it to our specific theorem.

Theorem 5.3.2 *Let E=$\{\alpha,\beta\}$ where α and β are permutations of a finite set X having k and l cycles, respectively. Suppose X contains an E-cycle. If $\gamma = \beta^{-1}\alpha$ has odd order, then k and l are odd.*

Proof. (of Theorem 5.3.2)
We need the following fact for this proof.

Lemma 5.3.3 *Let π be a permutation of n elements having r cycles. Then $sign(\pi) = (-1)^{n+r}$.*

Let P = {all elements of E-cycle from $\alpha(x)$} and Q = {all elements of E-cycle from $\beta(x)$}. Let $\delta = \beta^{-1}\pi$. So $\delta|P = \gamma|P$ and $\delta|Q = 1$. Therefore, because $order(\gamma)$ is odd, δ will have some odd cycles in common with γ (since $\delta|P = \gamma|P$) and also some 1-cycles (since $\delta|Q = 1$). Clearly, this causes $order(\delta)$ to be odd as well, so $sign(\delta) = 1$. Therefore, $sign(\pi) = sign(\beta)$ and by Lemma 5.3.3

$n + r \equiv n + l(mod\ 2)$. Set $r = 1$ because we only have one E-cycle and we have, $1 \equiv l(mod\ 2)$. Similarly, $1 \equiv k(mod\ 2)$, so, k and l are both odd. □

Theorem 5.3.1 is an immediate consequence of Theorem 5.3.2. Let $X = G$ and define $\alpha(x) = x\tau$ and $\beta(x) = x\sigma$. Then $\gamma(x) = x(\sigma^{-1}\tau)$ so γ has the same order as $\sigma^{-1}\tau$ and $k = index(\tau)$ and $l = index(\sigma)$. We have now established that if G is Hamiltonian and $order(\sigma^{-1}\tau)$ is odd, then $index(\sigma)$ and $index(\tau)$ are odd. Theorem 5.3.1 is the contrapositive of this. □

We used this theorem to classify a subset of our Cayley digraphs as non-Hamiltonian. Graphs for which this result did not apply were examined exhaustively.

5.3.2 DigHam

DigHam [20] is an algorithm developed by Brendan McKay to test for directed Hamiltonicity in (2,2)-diregular digraphs.

5.3.2.1 Algorithm

This algorithm maintains a data structure called *cycle* that classifies each arc as IN or OUT of the Hamilton cycle (initially $cycle[a] = DUNNO, \forall a \in A$). The method continues classifying arcs and propagating until either a Hamilton cycle is found or all possibilities have been checked. Propagation is explained with an example in Section 5.3.2.2. A full explanation is given as Algorithm 5.1.

5.3.2.2 Example

The propagation process is best explained with an example. Consider the digraph of Figure 5.2.

Assume we start at vertex X and set $cycle[(X, Z)] = OUT$. This forces $cycle[(X, Y)]$ to be IN. Further, since every vertex must have exactly one incoming arc in a Hamilton cycle, $cycle[(B, Z)]$ must be IN. Conversely,

Given (2,2)-diregular digraph G
begin
  for ***each*** $a \in A(G)$ **do**
    $cycle[a] = DUNNO$;
  endfor
  repeat
    Find $x \in V(G)$ where $cycle[(x,y)] = cycle[(x,z)] = DUNNO$;
    $cycle[(x,y)] = OUT$; Propagate;
    $cycle[(x,z)] = OUT$; Propagate;
  until *Hamilton cycle found or all possibilities are checked*;
end

Algorithm 5.1: Finding Hamilton Cycles in (2,2)-regular digraphs

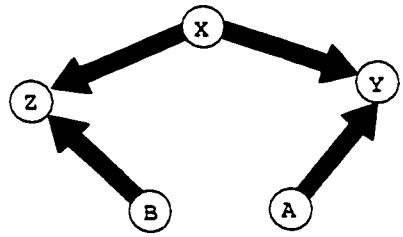


Figure 5.2: DigHam Algorithm Example

$cycle[(A,Y)]$ is forced OUT. This process is continued as long as possible. Arcs that are to be included in the cycle are drawn blue and arcs that are not included in the cycle are drawn red.

Notice that by using this propagation process we are still checking every *possible* Hamilton cycle. Therefore, the algorithm correctly determines Hamiltonicity of (2,2)-diregular digraphs.

5.3.3 Data

Directed Hamiltonicity characterization are part of the "non-isomorphic" and "non-color isomorphic" tables at

$$http://www.theory.csc.uvic.ca/\sim cos/cayley/.$$

This data will be very useful when we test for Hamiltonicity in cubic graphs in the next section.

5.4 Undirected Hamiltonicity

Many algorithms and heuristics have also been developed to find Hamilton cycles in undirected graphs [38]. A survey of these algorithms follows.

Pósa's algorithm attempts to extend the length of a partial path until a Hamilton cycle results. Note that no backtracking is ever done. If the algorithm cannot progress any further, a rotational transformation is done that allows for further extension of the path. The UHC algorithm of Angluin and Valiant is a slight modification of the Pósa technique. Instead of leaving edges in the graph once they have been used we remove them.

SparceHam is an algorithm that finds Hamilton cycles in sparse graphs in $\mathcal{O}(n^3 logn)$ time. Frieze's SparceHam method works by building up a cycle one edge at a time. If at any step there are no more valid edges to choose, we will rotate the built up cycle until there is a valid edge to choose. The Ham and HideHam algorithms use the same idea, with minor modifications.

Other algorithms such as DB2 by Brunacci, LongPath by Kocay and Li, FPGA-Ham [18] by Echo Liang, Ham by Donald Knuth and LinearHam by Thomason employ other techniques designed to find Hamilton cycles quickly for certain classes of graphs. We will describe the three algorithms we used to find the Hamilton cycles in this research in detail. They are CubHam by Brendan McKay, the Stone Carver's algorithm by Alex and Philipp Hertel, and Ian Shields' Pósa-like algorithm.

5.4.1 CubHam

Brendan McKay's CubHam program was used to test cubic Cayley graphs for Hamiltonicity. It takes advantage of the structure of cubic graphs to propagate decisions as far as possible in order to speed up the process of finding a Hamilton cycle. Note that if a graph is non-Hamiltonian cubham will have to search through all possibilities before it can classify the graph as non-Hamiltonian.

5.4.1.1 Algorithm

We maintain a data structure *cycle*, as in the DigHam algorithm, that contains classifications of edges as IN or OUT. The algorithm does a continuous process of propagation and classifying edges as OUT until it either finds a Hamilton cycle or has tried all possibilities. The process is listed in Algorithm 5.2.

5.4.1.2 Example

An example of the propagation process will make this algorithm more understandable. We will show the result of the algorithm on the 4-vertex complete graph, K_4, as shown in Figure 5.3(a). The process will start by calling $hamNode(0)$. For each of edges (0,1), (0,2) and (0,3) the algorithm will classify the edge as OUT and see if a cycle results from the propagation process. Consider setting $cycle[(0,1)]$ to OUT as shown in Figure 5.3(b) in red. This forces the other edges incident to vertices 0 and 1 (in blue) to be in the cycle. In this case, the 4 edges classified as being in the cycle make up a Hamilton cycle, so we are done.

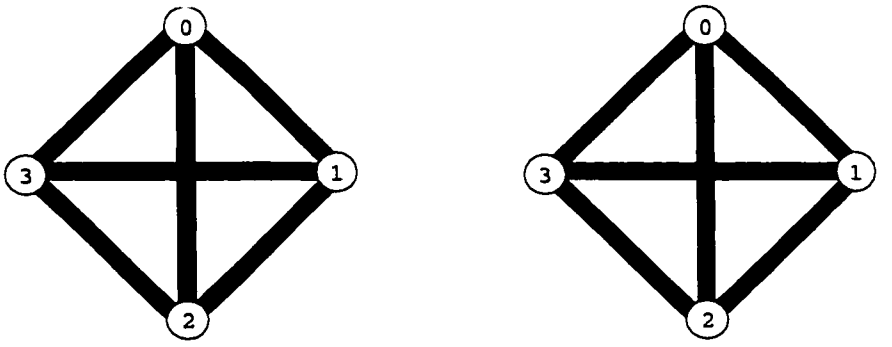
Notice that edge (2,3) could also be classified as OUT since there are two edges incident to each of vertices 2 and 3 that are classified IN.

```
Given cubic graph G
begin
    for each e ∈ E(G) do
        cycle[e] = DUNNO;
    endfor
    hamNode(0);
end
hamNode(v) begin
    Propagate; e1, e2, e3 ∈ E(G) are incident to v;
    if ≤ 1 of cycle[ei] are NO for 1 ≤ i ≤ 3 then
        for each ei where cycle[e] = NO or DUNNO do
            cycle[e] = NO; return hamNode(v + 1);
        endfor
    else
        return FALSE;
    endif
end
```

Algorithm 5.2: Finding Hamilton Cycles in Cubic Graphs

5.4.2 Stone Carver's Algorithm

A second method used to determine Hamiltonicity of cubic Cayley graphs was developed by Alex and Philipp Hertel [15], undergraduate students at the University of Victoria. Their algorithm is loosely based on the SparceHam technique introduced by Frieze. However, the Stone Carver's Algorithm works in the opposite way of SparceHam, it deletes edges from the graph until all that is left is a Hamilton cycle. Also, the algorithm is randomized, so, if it does not find a Hamilton cycle in a specified amount of time we cannot be sure that the graph is non-Hamiltonian.



(a) Original Graph (b) Hamilton Cycle

Figure 5.3: Cubham Algorithm Example

5.4.2.1 Algorithm

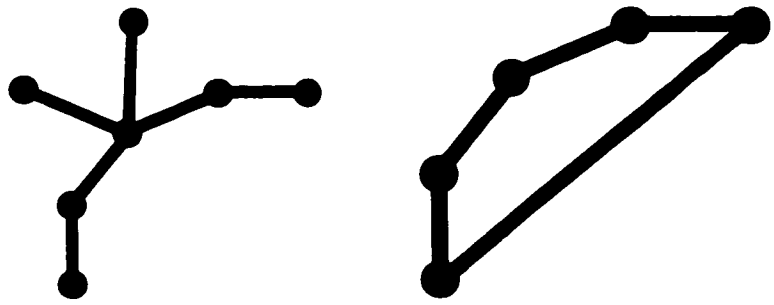
The basic method is documented as Algorithm 5.3.

```
Given graph G
begin
    while |E(G)| > |V(G)| do
        Remove an edge at random from G;
        Prune G using lightning and headhunter;
        if G is a cycle of length |V(G)| then
            return TRUE;
        endif
    endwhile
    return UNKNOWN;
end
```

Algorithm 5.3: Stone Carver's Algorithm

There are two pruning operations. The first of these is similar to the propagation used by Brendan McKay's Cubham algorithm. If there is a length 4 path that is forced to be in the Hamilton cycle, say *abcde*, then all other edges adjacent to vertex c cannot be in the cycle. This operation is called *lightning*

(see Figure 5.4(a)) and can be done in constant time. The second pruning operation is called *headhunter* (see Figure 5.4(b)). Define a *chain* to be a longest path currently in the Hamilton cycle. A *head* is an endpoint of a chain (each chain has exactly two heads). We can then be sure that the edge between the heads of a chain is not in the cycle (unless there is only one chain in the graph and it has length n-1). The headhunter operation can also be accomplished in constant time.



(a) Lightning (b) Headhunter

Figure 5.4: Stone Carver's Algorithm Pruning Techniques

At the first step (when no edges have been cut), we will choose a random edge from the set of all edges. However, at subsequent calls, we will choose a random edge to cut from the set of edges adjacent to a head. This will obviously increase the chance of finding a Hamilton cycle.

This heuristic will not find a Hamilton cycle in all Hamiltonian graphs, however, a second version was developed to solve this problem. The algorithm above is modified so that possibilities of cut edges are never tried more than once. If all sets of cut edges have been tried to no avail, we can be sure the graph is non-Hamiltonian.

5.4.2.2 Complexity

Through empirical tests on random regular graphs, this algorithm appears to run in $\mathcal{O}(nm)$ time where n is the number of vertices and m is the number of edges in the original graph. No proof of this bound is known.

5.4.3 Ian Shields' Pósa-Like Algorithm

A third algorithm used to find Hamilton cycles in cubic Cayley graphs is based on PhD research by Ian Shields [35] and a method by Basil Vandegriend [38]. The algorithm builds upon the Pósa algorithm described earlier. The details are currently unpublished.

5.4.4 Data

Data can be found under the heading "Cubic Cayley Graphs" at

$$http://www.theory.csc.uvic.ca/\sim cos/cayley/.$$

Of greatest importance is the column labeled "Hamiltonian?" as this characterizes each cubic Cayley graph as Hamiltonian or non-Hamiltonian and gives a cycle if one has been found. All but one of the class 1 cubic Cayley graphs G where $|V(G)| \leq 40320$ and $n \leq 9$ have had a Hamilton cycle found for them. A Hamilton cycle has been found for each class 2 cubic Cayley graph G where $|V(G)| \leq 2520$ and $n \leq 7$.

5.5 Measures of Difficulty

To better understand why some graphs give Hamilton cycle algorithms problems, we studied some measures of difficulty. This way, we can try to predict how long it may take to find a cycle using a certain algorithm.

5.5.1 Diameter

The *diameter* of a graph G is defined as the maximum length of all shortest paths between pairs of vertices in G. The diameter is often involved in running time bounds of an algorithm. For this reason, we computed the diameter of all Cayley graphs generated in order to produce an informal measure of how *hard* the graph is. In most cases, the graphs in which CubHam, the Stone Carver's algorithm and other backtracking programs had difficulty with were the ones with relatively high diameter.

5.5.1.1 Algorithm

For Cayley graphs, computing diameter is fairly trivial. Simply compute the shortest path between the first vertex (the identity) and every other vertex.

In a *breadth first search* (BFS) we explore vertices adjacent to the least recently discovered vertex that has unexplored edges. That is, extremes are searched last. A *breadth first search numbering* results from labeling the vertices in the order they are visited in the breadth first search. Notice that the shortest path between two vertices can be computed using a breadth first search. Recall that the diameter is the length of the longest shortest path.

```
Given Cayley graph G
begin
    Relabel the vertices of G using a BFS numbering;
    return( length of the shortest path from the first vertex to the
    |V(G)|^th vertex in G);
end
```

Algorithm 5.4: Computing the Diameter of a Cayley Graph

5.5.1.2 Example

As an illustration, for $|V(G)| = 2520$, the two graphs CubHam had problems with were $Cay(\{(0123)(45), (34)(56)\})$ and $Cay(\{(03)(1425), (34)(56)\})$. CubHam was of no help in classifying these graphs and therefore we eventually used Ian Shields' heuristic to find their Hamilton cycles. Both graphs have diameter of 17, which is significantly higher than the other 2520-vertex cubic Cayley graphs. Intuitively, it will take the backtracking structure of the CubHam algorithm longer to detect an impossibility since there is a relatively long shortest path. Of course, this is not a perfect measure, but it gives an idea of the difficulty that CubHam and other backtracking algorithms will have with the graph.

5.5.1.3 Correctness and Complexity

We can easily prove correctness of the diameter algorithm using some facts about Cayley graphs.

Theorem 5.5.1 *Given Cayley graph G, Algorithm 5.4 computes the diameter of G in $\mathcal{O}(|V(G)|)$ time.*

Proof. Since Cayley graphs are vertex transitive it therefore suffices to check paths starting at any vertex. Further, the distance to the vertex last visited by a breadth first search is by definition the diameter. This algorithm runs in $\mathcal{O}(|V(G)|)$ time since the breadth first search visits each node exactly once and the length of the path to the vertex visited last can be maintained dynamically. □

5.5.2 Shortest Odd Cycle

An *odd cycle* is a cycle that contains an odd number of vertices. A graph is *bipartite* if its vertices can be given one of two colors such that adjacent

vertices receive different colors. Another indicator of the difficulty of a graph is the length of its shortest odd cycle. Notice that if there are no odd cycles, then the graph is bipartite.

5.5.2.1 Algorithm

The length of the shortest odd cycle can be computed using a modified breadth first search at each vertex. For Cayley graphs, this can be simplified to performing a BFS at the first vertex (the identity) since the graphs are vertex transitive. The process is outlined in Algorithm 5.5.

```
Given Cayley graph G
begin
   for each vertex of G traversed in BFS order do
      if a vertex is repeated and the cycle through the repeated vertex
      has odd length, x then
         return x;
      endif
   endfor
   return NONE (i.e., bipartite);
end
```

Algorithm 5.5: Finding the Length of the Shortest Odd Cycle

5.5.2.2 Example

As for graphs with relatively large diameter, cubic Cayley graphs with a relatively large shortest odd cycle length were often the most difficult to find Hamilton cycles for. As an example, consider $Cay(\{(34)(56), (03)(1425)\})$. This graph has a shortest odd cycle length of 27; the longest among the 2520-vertex cubic Cayley graphs. As explained in the section on diameter CubHam has trouble finding a Hamilton cycle for this graph.

5.5.2.3 Correctness

The proof of correctness of this algorithm was first documented by Horton [27]. The original proof has been modified here for vertex transitive digraphs and therefore applies to Cayley graphs.

Theorem 5.5.2 *Algorithm 5.5 finds the shortest odd cycle of a Cayley graph.*

Proof. The minimum weight cycle basis of a vertex transitive digraph is a subset of the cycles created by adding each non-tree edge to the BFS tree at any vertex (because the graph is vertex transitive). There must be at least one odd cycle, since the sum of two even cycles is even. The first cycle that is found is independent of the other cycles chosen, so it must be part of the basis. Therefore, the minimum weight cycle basis must contain at least one minimum weight odd cycle. □

5.5.2.4 Complexity

The running time of the algorithm to find the shortest odd cycle was first documented by Monien [19]. A corollary has been added for the running time of the Cayley graph version of Algorithm 5.5.

Theorem 5.5.3 *The shortest odd cycle in an undirected, un-weighted graph G can be computed in time $\mathcal{O}(|V(G)| \cdot |E(G)|)$.*

Proof. From each vertex we can traverse each edge at most once and the work done at each edge is constant. □

Corollary 5.5.4 *The shortest odd cycle in an undirected, un-weighted Cayley graph G can be computed in time $\mathcal{O}(|E(G)|)$.*

Proof. We need only check one vertex since Cayley graphs are vertex transitive. □

5.6 Resolving Difficult Graphs

Often, instead of tackling an entire problem, we break the problem into smaller parts. In the case of Cayley graphs this can allow us to take advantage of symmetries, thus reducing the total amount of computation needed. This is exactly what was done with $Cay(\{(0123)(45), (34)(56)\})$ for which it was difficult to find a Hamilton cycle using CubHam alone.

We broke this graph down into two cases of possible Hamilton cycles. Case 1 includes all edges of the 1-factor (spanning 1-regular subgraph) induced by the involution $\tau = (34)(56)$. It turns out that the Hamilton cycle we eventually found resulted from this case. If there is no Hamilton cycle that passes through this 1-factor, then, somewhere on a Hamilton cycle there must be a $\tau\sigma\sigma\sigma\tau$ sequence as shown in Figure 5.5 (σ edges are red).

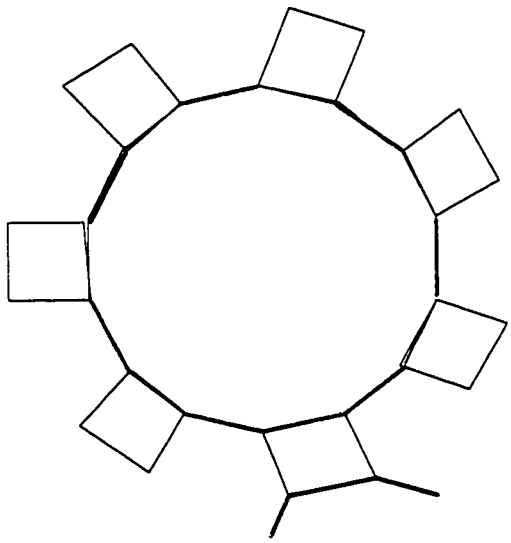
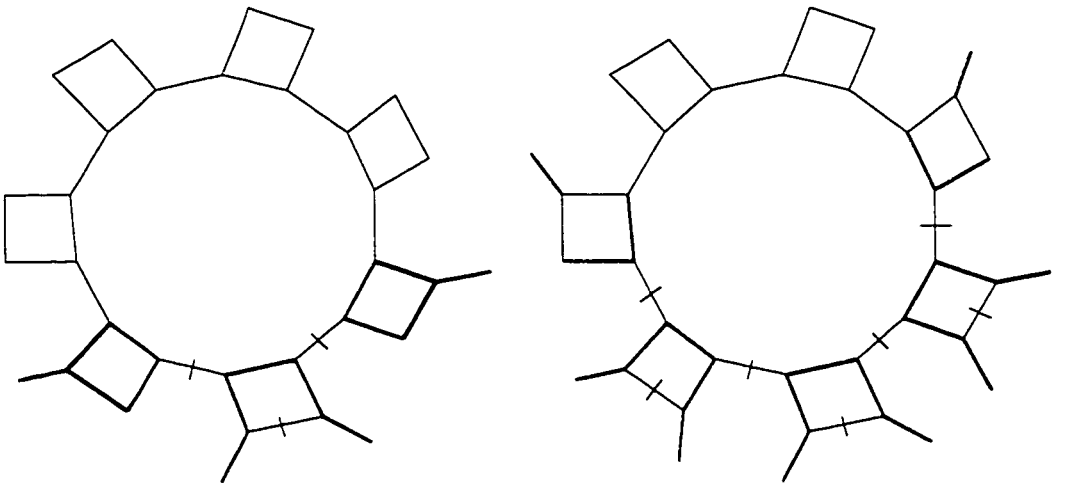


Figure 5.5: No Case 1 Hamilton Cycles in $Cay(\{(0123)(45), (34)(56)\})$

This induces two edges on two distinct 4-cycles as in Figure 5.6(a) (green edges). There are 4 ways to choose one edge from each side. These correspond to the 4 sub-cases which were tested. One such sub-case is shown in Figure 5.6(b).

A Hamilton cycle for this graph was first found by Ian Shields using his



(a) Overall (b) Subcase

Figure 5.6: Case 2 of $Cay(\{(0123)(45), (34)(56)\})$

Pósa-like algorithm, and later confirmed using Brendan McKay's CubHam algorithm.

5.7 Results

A survey by Witte and Gallian [40] as well as a survey of Curran and Gallian [8] given nice overviews of the known Hamiltonicity results for Cayley graphs. There are many classes of Cayley graphs that are provably Hamiltonian. Some interesting results were derived from our exploration of the Hamiltonicity of Cayley graphs. These include general properties of Cayley graphs as well as some classes of Cayley graphs that are provably Hamiltonian.

5.7.1 General Properties

From our data, we were able to derive some general classifications that shed light on which Cayley graphs are Hamiltonian.

Lemma 5.7.1 *Cubic Cayley graphs must have an even number of vertices.*

Proof. This result is true in general for k-regular graphs where k is odd. There are $\frac{kn}{2}$ edges in a k-regular graph. Since this number must be an integer and k is odd, n must be even. □

5.7.2 Classes of Hamiltonian Graphs

It is possible to prove that some classes of Cayley graphs are Hamiltonian. As a simple example, consider the Cayley graph $Cay(\{(12), (23), (34), \ldots, (n-1, n)\} : \mathbb{S}_n)$. The well known permutation generation algorithm of Steinhaus, Johnson and Trotter defines a Hamilton cycle for this Cayley graph since we are dealing with pairs of generator elements [34]. Therefore, this Cayley graph is Hamiltonian. This result however, is not very useful in our context since we are interested in 2-generated and cubic Cayley graphs only. For some 2-generated Cayley graphs we can construct a Hamilton cycle, thereby proving that these graphs are Hamiltonian.

5.7.2.1 Cycle Graphs

Some classes of Cayley generators produce the graph that is itself a cycle, C_n. These Cayley graphs are trivially Hamiltonian since the graphs themselves are Hamilton cycles.

Lemma 5.7.2 $\overrightarrow{Cay}(\{(12\cdots n), (n\cdots 21)\})$ *is Hamiltonian.*

Proof. This is an n-vertex 2-regular connected graph. Obviously, it is Hamiltonian. □

Lemma 5.7.3 $\overrightarrow{Cay}(\{\sigma, \tau\})$ *where* $\sigma \neq \tau$ *and* $order(\sigma) = order(\tau) = 2$ *is Hamiltonian.*

Proof. These generators obviously force a 2-regular connected graph, which trivially has a Hamilton cycle. □

5.7.2.2 General Graphs

Often, it is difficult to prove Hamiltonicity of general Cayley graphs. Below is a sample of results that were derived from our data and then proved mathematically. We noticed many classes of Cayley graphs that appear to be directed Hamiltonian, but realized that only a fraction of these appear to be provable.

Lemma 5.7.4 $\overrightarrow{Cay}(\{(12\cdots(n-1)),(23\cdots n)\})$ *is Hamiltonian.*

Proof. It can be easily verified that the cycle is $(\sigma^{n-2}\tau)^{n-2}$. □

The following results deal with generators that are distinct from each other.

Theorem 5.7.5 $\overrightarrow{Cay}(\{(12\cdots m),(m+1,m+2,\ldots,m+n)\})$ *is Hamiltonian if* $m|(m-1)n$ *or* $n|(n-1)m$.

Proof. It can be easily verified that the Hamilton cycle is $(\sigma^{m-1}\tau)^n$. This is a consequence of a result of Erdös and Trotter [37]. □

Corollary 5.7.6 $Cay(\{(12\cdots m),(m+1,m+2,\ldots,m+n)\})$ *is Hamiltonian.*

Proof. Similar to the cycle of the proof of Theorem 5.7.5 with cycle traversals going backwards if necessary. This Cayley graph is the Cartesian product of two Cayley graphs that are themselves cycles. It is well known [37] that the Cartesian product of two graphs that are Hamiltonian is also Hamiltonian, therefore, this Cayley graph is Hamiltonian. □

5.7.3 Empirical Results

From our data, it looks as though Conjecture 5.1.1 is true. However, all we can say for sure is the following.

Theorem 5.7.7 *All class 1 cubic Cayley graphs G where $n \leq 9$ and $|V(G)| \leq 40320$ are Hamiltonian (except $Cay(\{(12)(34)(56)(78), (013)(245)(78)\})$ which currently has unknown Hamiltonicity).*

Proof. By exhaustive computer search using Brendan McKay's cubham program, Ian Shields' Pósa-like algorithm and Alex and Phillip Hertel's Stone Carver's heuristic. □

Theorem 5.7.8 *All class 2 cubic Cayley graphs G where $n \leq 7$ are Hamiltonian.*

Proof. By exhaustive computer search. □

Chapter 6

Conclusion

6.1 Overview

Overall, it can be said that although Cayley graphs are easy to describe, it can be very difficult to enumerate and test them for various graph properties. We have found that among these properties, Hamiltonicity seems to be the most difficult and therefore the most interesting. Even if a graph is suspected as being non-Hamiltonian, it can be very difficult to prove it. Therefore, although there are a few graphs that we cannot find cycles for, we cannot be sure that they are non-Hamiltonian. In order to classify these graphs as non-Hamiltonian we must dissect them to find out why the graph cannot be Hamiltonian. As yet, we have not been able to do this for any of the graphs for which the Hamiltonicity is unknown.

6.2 Future Work

The future work in this area can be divided into two main categories: empirical and mathematical.

6.2.1 Future Empirical Work

It would be beneficial to generate Cayley graphs where $n > 9$ and $|X| > 2$ (and 2-generated cubic Cayley graphs for n=9). We stopped generating Cayley graphs where we did since the number of such graphs increases at an alarming rate.

As for isomorphism and color isomorphism, these results could be easily extended with enough computing power beyond the current vertex limit of 1000.

Along the way, we tested many pairs of Cayley graphs where $|V(G)| > 1000$ for isomorphism in order to cut down the number of graphs that needed to be tested for Hamiltonicity. However, a comprehensive test was never completed.

To date, we have tested almost all the cubic Cayley graphs that have been generated for directed and undirected Hamiltonicity. However, a few holes do exist, especially for directed Hamiltonicity. Also, initially we started testing for undirected Hamiltonicity on only cubic Cayley graphs since we wanted to use the CubHam Algorithm. However, more recently, we have been using the Stone Carver's Algorithm which handles any undirected graph. Therefore, we could test the 2-regular and 4-regular 2-generated Cayley graphs for Hamiltonicity using this algorithm.

6.2.2 Future Mathematical Work

Our formulae for the enumeration of 2-generated Cayley pairs and cubic Cayley sets have been simplified greatly. However, there remains a possibility that these formulae can be further simplified.

Not much is known about the enumeration of non-isomorphic and non-color isomorphic Cayley graphs. Results in this area are mostly simple upper bounds based on general graph enumeration. Also, further results about when two Cayley graphs are isomorphic and color isomorphic would be interesting.

With respect to the Hamiltonicity of Cayley graphs, more results pertaining to classes of Cayley graphs that are Hamiltonian would be interesting. Further to this, a proof to Conjecture 5.1.1 would be a remarkable innovation in this field.

Bibliography

1. S.B. Akers and B. Krishnamurthy, *A Group-Theoretic Model for Symmetric Interconnection Networks.* IEEE Trans. Comput., vol38, pages 555-565.

2. Bruce Arden and Wendy Tang, *Representations and Routing of Borel Cayley Graphs.* Technical Report, University of Rochester, New York, 1989.

3. Bruce Arden and Wendy Tang, *Representations and Routing of Cayley Graphs.* IEEE Transactions on Communications, 39:11 (1991), pages 1533-1537.

4. Eric Babson and Itai Benjamini, *Cut Sets and Normed Cohomology with Applications to Percolation.* Proc. Amer. Math. Soc., to appear.

5. N. Biggs, *Algebraic Graph Theory.* Cambridge University Press, 1974.

6. I.Z. Bouwer, W.W. Chernoff, B. Monson and Z. Star, *The Foster Census.* Charles Babbage Research Centre, 1988.

7. Daniel Cohen, *Basic Techniques of Combinatorial Theory.* Wiley, 1978.

8. Stephen Curran and Joseph Gallian, *Hamiltonian Cycles and Path in Cayley Graphs and Digraphs - A Survey.* Discrete Mathematics, 156 (1996), pages 1-18.

9. Eugene Curtin, *Cubic Cayley Graphs With Small Diameter.* Discrete Mathematics and Theoretical Computer Science, 4 (2001), pages 123-132.

10. Michael Fellows and Rodney Downey, *Parameterized Complexity.* Springer, 1999.

11. Joseph Gallian, *Contemporary Abstract Algebra.* D.C. Heath and Company, 1986.

12. Michael Garey and David Johnson, *Computers and Intractability.* W.H. Freeman & Company, 1979.

13. Israel Grossman and Wilhelm Magnus, *Groups and Their Graphs.* The Matheomatical Association of America, 1964.

14. Frank Harary and Edgar Palmer, *Graphical Enumeration.* Academic Press, 1973.

15. Alex Hertel and Philipp Hertel, *The Stone Carver's Algorithm.* Talk given at UVic Computer Science Department, October 5, 2001.

16. Alex Hertel and Philipp Hertel, *The Stone Carver's Hamiltonian Cycle Algorithm.* to appear, 2002.

17. Donald E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching.* Second Edition, Addison-Wesley, 1998.

18. Echo Liang, *FPGA Algorithm for Finding Hamilton Cycles in Undirected Graphs.* Personal Conversation, November 21, 2001.

19. B. Monien, *The Complexity of Determining a Shortest Cycle of Even Length.* Computing, 31 (1983), pages 355-369.

20. Brendan McKay, *DigHam Algorithm.* C program.

21. Brendan McKay, *Practical Graph Isomorphism.* Congressus Numerantium, 30 (1981), pages 45-87.

22. Brendan McKay and Cheryl Praeger, *Vertex-Transitive Graphs Which Are Not Cayley Graphs, I.* Journal of Australian Mathematics Society (Series A), 56 (1994), pages 53-63.

23. Brendan McKay and Gordon Royle, *The Transitive Graphs With At Most 26 Vertices.* ARS Combinatoria, 30 (1990), pages 161-176.

24. Brendan McKay and Gordon Royle, *Cubic Transitive Graphs.* http://www.cs.uwa.edu.au/ gordon/remote/cubtrans/index.html, 1996.

25. Brendan McKay and Frank Ruskey, *On the Hamiltonicity of 2-generated Cayley graphs: the world's longest Hamilton cycle.* to appear.

26. Mirka Miller, Joan Gimbert, Frank Ruskey and Joseph Ryan, *Iterations of Eccentric Digraphs.* to appear, 2002.

27. Wendy Myrvold, *Shortest Odd Cycle.* Personal communication, August 13, 2001.

28. Wendy Myrvold and Frank Ruskey, *Ranking and Unranking Permutations in Linear Time.* Information Processing Letters, 79 (2001), pages 281-284.

29. M.V.S. Ramanath and T.R. Walsh, *Enumeration and Generation of a Class of Regular Digraphs.* Journal of Graph Theory, 11:4 (1987), pages 471-479.

30. R. Rankin, *A Campanological Problem in Group Theory.* Proc. Camb. Phil. Soc., 44 (1948), pages 17-25.

31. Zsuzanna Rocca, *Cellular Automata on Cayley Graphs.* PhD Dissertation, ENS Lyon, France, 1994.

32. Frank Ruskey, *Combinatorial Generation.* Working Version, 1995-2001.

33. Frank Ruskey, *Combinatorial Object Server.* http://www.theory.csc.uvic.ca/~cos.

34. Frank Ruskey and Carla Savage, *Hamilton Cycles That Extend Transposition Matchings in Cayley Graphs of* $\mathbb{S}_n$. SIAM Journal of Discrete Mathematics, volume 6, number 1, pages 152-166.

35. Ian Shields and Carla Savage, *A Hamilton Path Heuristic with Applications to the Middle Two Levels Problem.* to appear.

36. Richard Swan, *A Simple Proof of Rankin's Campanological Theorem.* American Mathematics Monthly, 1999, pages 159-161.

37. William Trotter and Paul Erdös, *When the Cartesian Product of Directed Cycles is Hamiltonian* Journal of Graph Theory, volume 2 (1978), pages 137-142.

38. Basil Vandegriend, *Finding Hamiltonian Cycles: Algorithms, Graphs and Performance.* University of Alberta Master's Thesis, 1998.

39. Douglas West, *Introduction to Graph Theory.* Prentice Hall, Second Edition, 2001.

40. David Witte and Joseph Gallian, *A Survey: Hamilton Cycles in Cayley Graphs.* Discrete Mathematics, 51 (1984), pages 293-304.

Appendix A

List of Symbols

Below is a list of all standard mathematical symbols used in this thesis, in order of appearance.

$V(G)$: Vertex set of graph (digraph) G

$E(G)$: Edge set of graph G

$A(G)$: Arc set of digraph G

$d(v)$: Degree of vertex v

$d^+(v)$: Out-degree of vertex v

$d^-(v)$: In-degree of vertex v

$|S|$: Size of set S

X: Generating set

n: Number of generator symbols

$\overrightarrow{Cay}(X : G)$: Cayley Digraph

$Cay(X : G)$: Cayley Graph

$e(u)$: Eccentricity of vertex u

$ED(G)$: Eccentric digraph of G

$p(n)$: Numerical partitions of n

$\mu(n)$: The *Möbius* function

$PC(\alpha)$: The pseudo-centralizer of α

$h(a)$: Number of permutations with cycle structure a

ϕ: Euler Phi function

$gcd(r, t)$: Greatest common divisor of integers r and t

$lcm(r,t)$: Least common multiple of integers r and t

$rank(\pi)$: Rank of π

$unrank(r)$: Element with rank r

$index(\sigma)$: Index of σ

Below is a list of all non-standard symbols defined in this thesis, in order of appearance.

P_n: Number of generator pairs

I_n: Number of length n involutions

$I_{2,n}$: Number of non-isomorphic Cayley generator pairs on n symbols

$F_{2,n}$: Number of non-isomorphic Cayley generator pairs with no common fixed point on n symbols

$I'_{2,n}$: Number of non-isomorphic Cayley generator pairs on n symbols where exactly one generator is an involution

$F'_{2,n}$: Number of non-isomorphic Cayley generator pairs with no common fixed point on n symbols where exactly one generator is an involution

$I'_{3,n}$: Number of non-isomorphic Cayley generator triples on n symbols where all generators are involutions

$F'_{3,n}$: Number of non-isomorphic Cayley generator triples with no common fixed point on n symbols where all generators are involutions

$P(\lambda)$: Number of permutations with cycle structure λ

$li(G)$: Minimum number of generator symbols needed to generate any Cayley graph isomorphic to G

d_p: Number of digraphs with p vertices

c_p: Number of connected digraphs with p vertices

$d_p^{2,2}$: Number of (2,2)-diregular digraphs with p vertices

$c_p^{2,2}$: Number of (2,2)-diregular connected digraphs with p vertices

$lci(G)$: Minimum number of generator symbols needed to generate any Cayley graph color isomorphic to G

$C_{2,k}^{i}$: Number of non-isomorphic 2-generated Cayley graphs on k vertices

$C_{2,k}^{ci}$: Number of non-color isomorphic 2-generated Cayley graphs on k vertices

Appendix B

List of Programs

Below is a list of all programs used to generate and test Cayley graphs for various properties in this thesis.

genall: Generates all non-isomorphic Cayley generator pairs
Developed by Frank Ruskey

3genall: Generates all non-isomorphic Cayley generator triples
Developed by Scott Effler (based on genall program)

gengroup: Converts a generator pair into an adjacency list
Developed by Frank Ruskey

3gengroup: Converts a generator triple into an adjacency list
Developed by Scott Effler (based on gengroup program)

Nauty: Tests graph isomorphism
Developed by Brendan McKay

checkColorIso: Tests Cayley graphs for graph color isomorphism
Developed by Scott Effler

digham: Finds directed Hamilton cycles in (2,2)-diregular digraphs
Developed by Brendan McKay

cubham: Finds Hamilton cycles in 3-regular graphs
Developed by Brendan McKay

diameter: Determines the diameter of a Cayley graph
Developed by Scott Effler

oddCycle: Determines the length of the shortest odd cycle in a Cayley graph and subsequently determines whether or not a Cayley

graph is bipartite
Developed by Scott Effler

Shields' Program: Finds Hamilton cycles in Cayley graphs
Developed by Ian Shields' (extension of an algorithm by Basil Vandegriend)

Stone Carver's Algorithm: Finds Hamilton cycles in undirected graphs
Developed by Alex and Philipp Hertel

EccCubic: Finds the length of the shortest eccentric cycle in a cubic graph
Developed by Scott Effler and Frank Ruskey

In addition, many simple perl scripts and C and Java programs were used to obtain intermediate results and to generate the tables of data seen in this paper and on the data website.