HAMILTONICITY OF CERTAIN VERTEX SYMMETRIC GRAPHS

by

Ming Jiang

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the Department of
Computer Science

ACCEPTED

[redacted]

We accept this thesis as conforming
to the required standard

[redacted]

Dr. F. Ruskey, Supervisor, Dept. of Computer Science.

[redacted]

Dr. W.J. Myrvold, Departmental Member, Dept. of Computer Science

[redacted]

Dr. C.R. Miers, Outside Member, Dept. of Maths. and Statistics

[redacted]

Dr. G. MacGillivray, External Examiner, Dept. of Maths. and Statistics

Supervisor: Dr. F. Ruskey

Abstract

Vertex symmetric graphs play an important role in the design of parallel architectures. In this thesis, we study Hamiltonicity in specific families of Cayley graphs and vertex symmetric graphs. In the first part of the thesis, we study the Hamiltonicity in Faber-Moore digraphs. Faber-Moore digraphs have been proposed as a method of efficient parallel architectures in which information flows unidirectionally. We prove that every Faber-Moore digraph is Hamiltonian. In addition, we give a sufficient and necessary condition for the existence of Hamilton path between any pair of vertices. In the second part of the thesis, we give an efficient algorithm to generate permutations by a tree of transpositions. This algorithm finds Hamilton paths in the corresponding tree Cayley graphs. We give some open problems in the last part of the thesis.

Examiners:

Dr. F. Ruskey, Supervisor, Dept. of Computer Science

Dr. W.J. Myrvold, Departmental Member, Dept. of Computer Science

Dr. C.R. Miers, Outside Member, Dept. of Maths. and Statistics

Dr. G. MacGillivray, External Examiner, Dept. of Maths. and Statistics

Contents

List of Figures

Acknowledgements

First I would like to thank my supervisor, Prof. Frank Ruskey, for his encouragement and support during my stay in Victoria. He allows me to work in a relatively independent way, while he always offers invaluable guidance.

I would also like to thank my officemates, Michael Dinneen and Anand Srinivasan, who gave me a lot of help in learning the intricacies of various software packages.

Finally, I would like to thank my wife Yanjing for her enthusiasm and her loving support.

Chapter 1

Introduction

In this thesis, we explore properties and algorithms about vertex symmetric graphs, including Cayley graphs. Cayley graphs are characterized by their high symmetry. The relationship between two global properties of graphs, symmetry and Hamiltonicity, attracts a lot of research in both graph theory and group theory. Also, Cayley graphs have applications in the design of parallel architectures. Graphs are natural tools for describing parallel architectures, in which vertices represent processors and edges (arcs) represent communication channels. So graph theory is a natural tool to study the combinatorial properties of parallel architectures. In Chapter 2 and 3, we prove that each member of a family of vertex symmetric digraphs, the Faber-Moore digraphs, is Hamiltonian. Faber-Moore digraphs have been proposed as efficient unidirectional parallel architectures. In Chapter 2, we study a special case of these digraphs, which provides a basis in the proof of general case in Chapter 3. In Chapter 4, we give an efficient algorithm for generating permutations by a tree of transpositions. This permutation generation problem is equivalent to finding a Hamilton path in the corresponding tree Cayley graphs. In the last chapter of this thesis, we give the conclusions and

some open problems.

In the rest of this chapter, we give some definitions from graph theory and algebra in Sections 1 and 2, discuss Cayley graphs (digraphs) and Hamiltonicity in Section 3 and applications of Cayley graphs (digraphs) in the design of parallel architectures in Section 4.

1.1 Graph theoretic preliminaries

In this section, we give graph theoretic definitions that are used in this thesis. They can be found in most graph theory books, such as Bondy and Murty[BoMu].

Definition: A **multigraph** $G = (V, E)$ consists of a set V whose elements are called **vertices**, and a multiset E, whose elements are unordered pairs of vertices in V, called **edges**. If $uv \in E$ then u and v are said to be adjacent. An edge of the form uu is called a loop. If E is a set (no repeated elements), then G is called a **graph**. A graph G is **simple** if it does not contain loops.

Definition: A **digraph** $G = (V, A)$ consists of a set V, whose elements are called vertices, and a set A, whose elements are ordered pairs of vertices in V, called **arcs**. If $uv \in A$, uv is said incident from u and incident to v.

Definition: The **order** of graph (digraph) $G = (V, E)$ is $|V|$, sometimes we denote it by $|G|$.

Definition: A **walk** in a graph (digraph) is a sequence of vertices $v_1, v_2, \ldots, v_n$ such that for all $1 \leq i < n$, $v_i v_{i+1}$) is an edge (arc) in G. A walk without repeated vertices is called a **path**. A path starting at u and ending at v is called a uv-path. The number of edges in a path is called the **length**

of the path. If the first vertex and the last vertex in a path are adjacent, then it is called a **cycle.** A path that contains every vertex in G is called a **Hamilton path** of G. Similarly, a **Hamilton cycle** of G is a cycle that contains every vertex of G. If G has a Hamilton cycle, it is **Hamiltonian.** If for any vertex $u, v \in V(G)$, there is a Hamilton path starting at u and ending at v, then G is **Hamilton-connected.**

Definition: A graph G is **connected** if there is a path between any two vertices in G; otherwise, it is **disconnected.** A digraph G is **strongly connected** if for each vertex u, there is a path from u to every other vertex in G.

Definition: A k-vertex cut in graph $G(V, E)$ is a subset V' of V such that $G - V'$ is empty or disconnected and $|V'| = k$. G is **k-connected** if G has a k-vertex cut but no $(k-1)$-vertex cut.

Definition: In a graph G, the **neighbourhood** of vertex v is $N(v) = \{u | uv \in E\}$, the **degree** of v is the number of elements in $N(v)$. A graph G is **k-regular** if every vertex in the graph has degree k. For a digraph, the **in-degree** of v is the number of arcs incident to v and the **out-degree** of v is the number of arcs incident from v. A digraph G is k-regular if every vertex in the graph has both in-degree and out-degree k.

Definition: The **distance** from u to v, $d(u, v)$, is the length of a shortest uv-path. The **diameter** of a connected graph (strongly connected digraph) $G = (V, E)$ is the least integer D such that for all vertices u and v in G, $d(u, v) \leq D$.

Definition: A graph (digraph) $G = (V, E)$ is **vertex symmetric** (or **vertex transitive**) if for each pair of vertices u and v in G, there exists a bijection $\theta : V \to V$ such that : (1) $\theta(u) = v$ and (2) $ab \in E$ implies

$\theta(a)\,\theta(b) \in E$. That is, θ preserves the adjacencies of G.

Definition: Two simple graphs G and H are **isomorphic** if and only if there is a bijection $\theta : V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $\theta(u)\,\theta(v) \in E(H)$.

In this thesis, unless specified otherwise, the graphs (digraphs) are simple and connected (strongly connected).

1.2 Algebraic preliminaries

This section contains some algebraic definitions that are used in this thesis. They can also be found in many algebra books, such as Fraleigh [Fr] and Gallion [Ga].

Definition: A **group** $(G, *)$ is a set G, together with a binary operation $*$ on G such that the following three properties are satisfied:

1. *Associativity*

   The binary operation $*$ is associative, that is, for all a, b, c in G, $(a * b) * c = a * (b * c)$.

2. *Identity*

   There is an element e in G such that $e * a = a * e = a$ for any element a in G. The element e is called the **identity** of G.

3. *Inverse*

   For each element a in G, there is an element a^{-1} in G such that $a * a^{-1} = a^{-1} * a = e$. The element a^{-1} is called the **inverse** of a.

A group G is called **abelian** if it is also commutative, that is for any elements a and b in G, $a * b = b * a$.

Definition: The **symmetric group** S_n is the collection of permutations of the set $[n] = \{1, 2, \ldots, n\}$ where group multiplication is defined by composition of permutations. If

$$\pi = \begin{pmatrix} 1234 \\ 2341 \end{pmatrix}, \quad \mu = \begin{pmatrix} 1234 \\ 3142 \end{pmatrix},$$

$$\pi\ \mu = \begin{pmatrix} 1234 \\ 1423 \end{pmatrix}$$

because

$$\begin{array}{ccccc} & \pi & & \mu & \\ 1 & \rightarrow & 2 & \rightarrow & 1 \\ 2 & \rightarrow & 3 & \rightarrow & 4 \\ 3 & \rightarrow & 4 & \rightarrow & 2 \\ 4 & \rightarrow & 1 & \rightarrow & 3. \end{array}$$

We often use one-line notation or cycle notation to represent permutations. In one-line notation, we use

$$a_1\ a_2\ \ldots\ a_n$$

to denote

$$\begin{pmatrix} 1\ 2 \ldots n \\ a_1 a_2 \ldots a_n \end{pmatrix}.$$

In cycle notation, for example, we use

$$(142)\,(35)$$

to denote

$$\begin{pmatrix} 1\,2\,3\,4\,5 \\ 4\,1\,5\,2\,3 \end{pmatrix}.$$

Permutations of the form (ij) are called **transpositions.**

Definition: The **cyclic group** is $(Z_n, \oplus)$, where $Z_n = \{0, 1, 2, \cdots, n-1\}$ and $\oplus$ is addition modulo n.

Definition: Let G be a finite group and S a subset of G. We define a digraph $Cay(S : G)$, called the **Cayley digraph** of G with subset S as follows:

1. Each element of G is a vertex of $Cay(S : G)$.
2. For x and y in G, there is an arc from x to y if and only if there exists some $s \in S$ such that $x * s = y$. We call this arc has **label** s.

If we also require $S = S \cup S^{-1}$ (S^{-1} is the set of inverses of elements in S) then $Cay(S : G)$ is undirected, so it is called a **Cayley graph.**

Figure 1.1 is an example of Cayley digraph.

We list a few commonly used Cayley graphs. The **n-cube** is the graph whose vertices are ordered n-tuples of 0's and 1's, two vertices are connected if and only if they differ in exactly one coordinate. Let $e_i = (\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{n-i})$. The n-cube is $Cay(\{e_i \mid 1 \leq i \leq n\} : (Z_2)^n)$. See Figure 1.2 for a 3-cube. The **n-pancake graph** is

$$Cay(\{(1,i)(2,i-1)\cdots(\lfloor i/2 \rfloor, \lceil i/2 \rceil) \mid 2 \leq i \leq n\} : S_n).$$

A **tree Cayley graph** is $Cay(T : S_n)$, in which T is a tree labeled with $\{1, 2, \ldots, n\}$, where two permutations are adjacent if and only if they differ

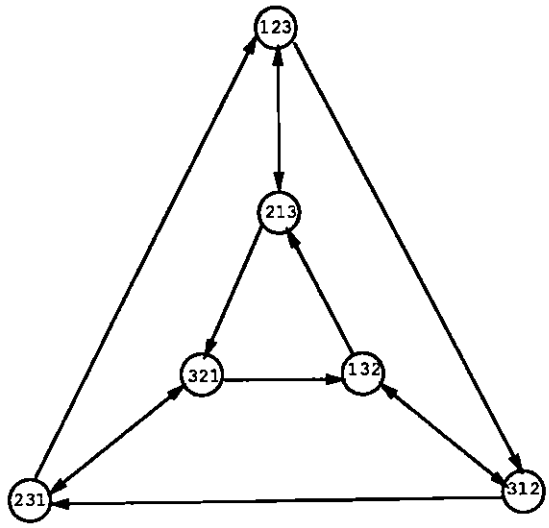


Figure 1.1: $Cay(\{(12),(321)\} : S_3)$.

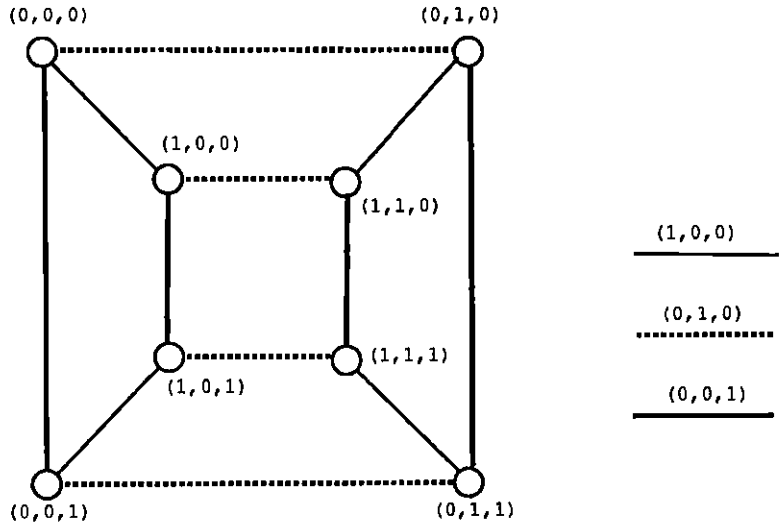


Figure 1.2: The 3-cube as a Cayley graph.

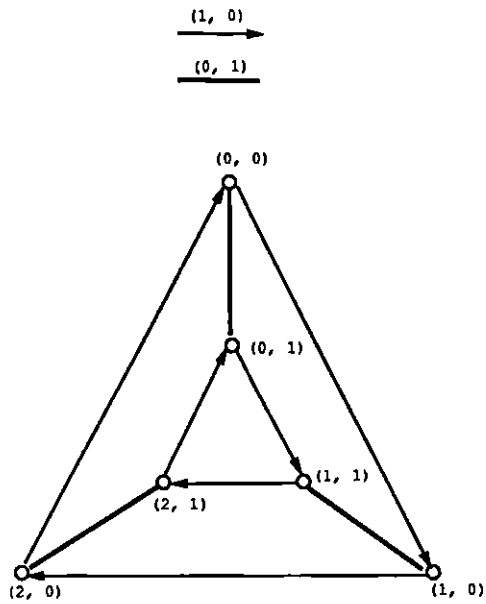


Figure 1.3: $Cay(\{(1,0),(0,1)\} : Z_3 \oplus Z_2)$, a strongly connected Cayley digraph that is not Hamiltonian.

by a transposition (ij), and ij is an edge in T. If T is a "star", all vertices except one have degree one, the corresponding tree Cayley graph is called a **star graph.** A 4-star graph can be found on Page 40.

1.3 Cayley graphs (digraphs) and Hamiltonicity

There is a lot of research about the Hamiltonicity (that is, the question of whether a graph has a Hamilton path or cycle or not) of Cayley graphs. This is largely due to two famous conjectures.

Conjecture 1 *(T.D.Parsons and others) There is a Hamilton cycle in every connected Cayley graph.*

The above conjecture is not true for Cayley digraphs in general. Figure 1.3 is $Cay(\{(1,0),(0,1)\} : Z_3 \oplus Z_2)$. It is strongly connected but has no Hamilton cycle.

A related conjecture is:

Conjecture 2 *(Lovász) There is a Hamilton path in every connected vertex symmetric graph.*

This conjecture is not true for Cayley digraphs in general, either. A counterexample can be found in [WiGa]. The difference between the two conjectures can be illustrated by the well-known Peterson graph. The Peterson graph is a vertex symmetric graph that is not a Cayley graph. It contains Hamilton paths but no Hamilton cycle [Gr].

There are some sufficient conditions or necessary conditions for whether a graph is Hamiltonian. They can be found in textbooks by Bondy and Murty [BoMu] and others. But no effective necessary and sufficient condition is known.

Theorem 1 *(Dirac) Let G be a simple graph of order $n(n \geq 3)$. If the minimum degree of G is not less than $n/2$, then G is Hamiltonian.*

Theorem 2 *(Ore) If G is a graph of order n such that for every pair of non-adjacent vertices x and y in G, $deg(x) + deg(y) \geq n$, then G is Hamiltonian.*

The above theorems show that any k-regular graph of order at most $2k$ is Hamiltonian. The following theorem goes a step further.

Theorem 3 *(Jackson) Every 2-connected k-regular graph with order not more than $3k$ is Hamiltonian [Ja].*

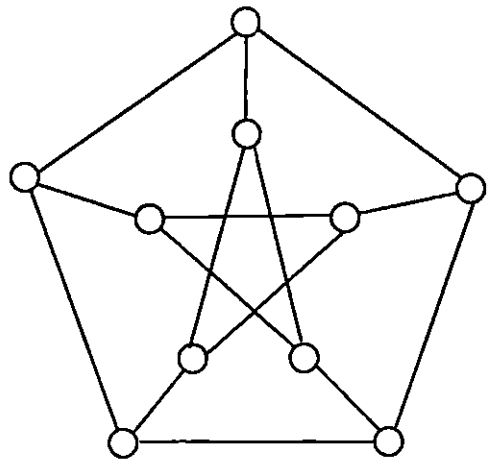


Figure 1.4: Peterson graph: a vertex symmetric graph that is not a Cayley graph.

A detailed survey of related theorems can be found in [Go]. Unfortunately these theorems can not be applied to Cayley graphs with low degrees, as do the graphs considered in this thesis. There is no known general theorem for deciding whether a Cayley graph is Hamiltonian. There are also some theorems that investigate the problem from the algebraic point of view.

Theorem 4 *Every connected Cayley graph of an abelian group of order at least three is Hamiltonian.*

The proof can be found in [Ga]. A stronger result is given by Chen and Quimpo [ChQu]. Readers who are interested in the progress in these two conjectures may refer to surveys [Go] [WiGa] and [Ma]

1.4 Cayley graphs (digraphs) and interconnection networks

Cayley graphs play an important role in design of interconnection networks. A process/communication interconnection network is often modeled as a graph (digraph), in which vertices correspond to processor/communication ports, and edges corresponds to communication channels. Symmetric interconnection networks are of special interests in network design because they have the property that the network viewed from any vertex looks the same when the corresponding graph is vertex symmetric. In such a network congestion problems are minimized since the load will be distributed through all the vertices uniformly. Moreover, this symmetry allows for identical processors at every vertex with identical routing algorithms [AkKr]. Cayley graphs are ideal for modeling symmetric interconnection networks, as described by the following well known theorem:

Theorem 5 *Every Cayley graph (digraph)* $Cay(S : G)$ *is vertex symmetric.*

Proof: Let a and b be any two elements in G, we define $\theta(x) = (ba^{-1})x$ for all $x \in G$.

1. θ is bijective. If $\theta(x_1) = \theta(x_2)$, that is $ba^{-1}x_1 = ba^{-1}x_2$, then

$$x_1 = (ab^{-1})(ba^{-1}x_2) = aea^{-1}x_2 = x_2.$$

Since G is finite and θ is a one-to-one mapping, it is also an onto mapping, and thus is a bijection.

2. θ preserves adjacencies. If x is adjacent with (to) y in the Cayley graph (digraph) then there is a $s \in S$ such that $xs = y$. $\theta(x)$ is also adjacent with (to) $\theta(y)$ since $(ba^{-1}x)s = ba^{-1}y$. □

Another reason that Cayley graphs are ideal for processor/communication architecture is because all proposed Cayley graphs are Hamiltonian. A sorting algorithm (a sequence together with a sequential algorithm) of a parallel architecture requires that we have a predetermined order in which the sorted data is to result. It can be shown that a necessary condition for the existence of a sorting algorithm is that this ordering forms a Hamilton path [AkHaKr].

Besides the n-cube, which is used as commercial parallel architecture, there are other possible alternatives for efficient parallel architectures, such as star graphs, pancake graphs, and cube-connected graphs etc [AkKr]. All of these architectures are based on undirected graphs, which models the situation where information can flow in both directions in the communication channels. In some cases, it is desirable to transfer information in one direction only. This situation can be modeled using directed graphs.

Faber and Moore [FaMo] studied, $\Gamma_\Delta(D)$, a family of large vertex symmetric digraphs. These digraphs may be defined as digraphs on $\Delta + 1$ alphabets in the following way: The vertices are labeled with different words of length D ($D \leq \Delta + 1$), $x_1 x_2 \ldots x_D$, each of which form a D-permutation of an alphabet of $\Delta + 1$ letters. The adjacency is given by:

$$x_1x_2\ldots x_D \rightarrow \begin{cases} x_{D+1}x_1x_2x_3\ldots x_{D-3}x_{D-2}x_{D-1}, & if\ x_{D+1} \neq x_1, x_2, \ldots, x_D \\ x_Dx_1x_2\ldots x_{D-3}x_{D-2}x_{D-1} \\ x_{D-1}x_1x_2\ldots x_{D-3}x_{D-2}x_D \\ x_{D-2}x_1x_2\ldots x_{D-3}x_{D-1}x_D \\ \ldots \\ x_2x_1x_3\ldots x_{D-2}x_{D-1}x_D \end{cases}$$

These digraphs have order $(\Delta+1)!/(\Delta-D+1)!$, diameter D and are Δ-regular [FaMo] [CoFi]. For convenience, we denote $\Gamma_\Delta(D)$ by $G(n,k)$ with $n=\Delta+1$, $k=D$ and $n \geq k$.

In Chapter 2, we will study the Hamiltonicity of $G(n,k)$ when $n=k$. In Chapter 3, we will study the Hamiltonicity of $G(n,k)$ in general.

Chapter 2

Right and Left Rotation Cayley Digraphs

In this chapter, we prove that the Faber-Moore digraphs $G(n,k)$ are Hamiltonian when $n = k$. Additionally, we give a sufficient and necessary condition for the existence of a Hamilton path between two vertices. For convenience, we denote $G(n,k)$ when $n = k$ by R_n, the right rotation Cayley digraph. See Figure 1.1 for R_3. Let r_i be the ith right rotation $(i\ i-1\ \cdots\ 2\ 1)$. The Cayley digraph R_n is $Cay(X_n : S_n)$ where $X_n = \{r_2, r_3, \cdots, r_n\}$, and S_n is the symmetric group on $[n]$. We think of the elements of S_n as being permutations $\pi = \pi(1)\pi(2)\cdots\pi(n)$ of $[n]$, and the elements of X_n as acting on these permutations as follows.

$$\pi(1)\pi(2)\cdots\pi(n)(i\ i-1\ \cdots\ 2\ 1) = \pi(i)\pi(1)\pi(2)\cdots\pi(i-1)\pi(i+1)\cdots\pi(n)$$

2.1 Hamilton paths and cycles in R_n

We begin by proving a necessary condition for the existence of Hamilton paths in R_n.

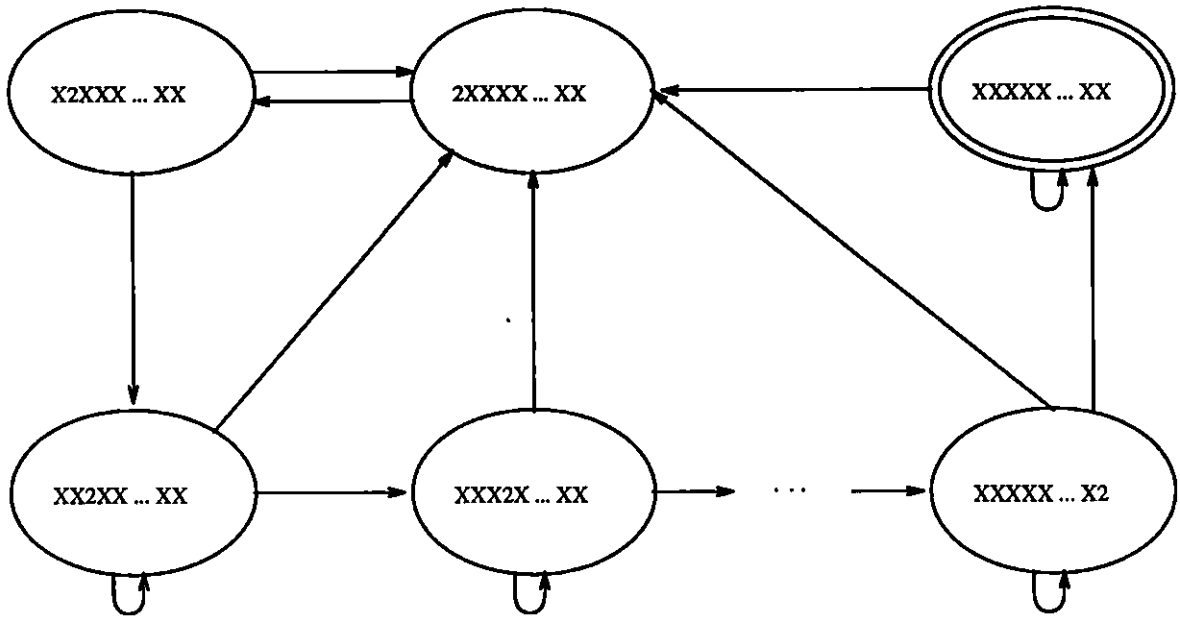


Figure 2.1: The multigraph $C_{n,k}$ obtained by condensing $G(n,k)$.

Lemma 1 *If there is a Hamilton path $\pi_1, \pi_2, \cdots, \pi_d$ in $G(n,k)$ from $\pi_1 = 12\cdots k$ to $\pi_d = x_1 x_2 \cdots x_k$, then $x_1 = 2$. ($d = C(n,k)\,k!$, the number of vertices in $G(n,k)$).*

Proof: Consider the directed multigraph $C_{n,k}$ formed by condensing, for $i = 0, 1, 2, \cdots, k$, into a single supervertex s_i all those permutations π for which $\pi(i) = 2$. The supervertex s_0 contains all the permutations that don't have element 2 in them. All the supervertices except s_0 contain $d' = C(n-1, k-1)(k-1)!$ permutations. For each arc $v_i v_j$ in $G(n,k)$, where v_i is in s_i and v_j is in s_j, there is an arc from s_i to s_j. See Figure 2.1 for $C_{n,k}$. A Hamilton path H starting at $12\cdots k$ in $G(n,k)$ becomes a walk W in $C_{n,k}$ starting at s_2. Let $d^-(s_i)$ denote the number of arcs in W of the form $s_j \rightarrow s_i$ for some j, and $d^+(s_i)$ the number of the form $s_i \rightarrow s_j$ for some j. Since each permutation occurs exactly once and the path starts at

$12\cdots k$, $d^-(s_2) = d' - 1$ and $d^-(s_1) = d'$. Observe that the only arcs entering s_2 are those from s_1 (since the only way that 2 can be brought into the second position is by applying a generator to a permutation with 2 in the first position) and all arcs leaving s_1 enter s_2. Thus, $d^+(s_1) = d' - 1$. But this means that the walk W must end at s_1. □

To prove a sufficient condition we will proceed by induction, but inductive hypothesis must be strengthened to be more than just the existence of a Hamilton cycle.

Definition: A (j,n)-pair in a path in R_n is a pair of successive permutations π and π' on the path such that the arc $\pi \to \pi'$ is labeled r_n and $\pi(n) = \pi'(1) = j$.

If there is a (j,n)-pair π and π', then $(n-1)!$ permutations with element j in position n can be inserted between π and π', as will shown in the proof of next lemma.

Lemma 2 *Let $n \geq 3$. For any permutation $x_2x_3\cdots x_n$ of $[n] \setminus \{2\}$, there is a Hamilton path H in R_n starting at $12\cdots n$ and ending at $2x_2x_3\cdots x_n$. Furthermore, the path H has a (j,n)-pair for each $j = 1,2,\cdots,n$.*

Proof: Our proof is by induction on n. For $n = 3$, there are two paths.
123 21$\underline{3}$ 3$\underline{21}$ $\underline{1}$32 31$\underline{2}$ $\underline{2}$31 , and 12$\underline{3}$ 3$\underline{12}$ $\underline{2}$31 32$\underline{1}$ $\underline{1}$32 213.
The $(j,3)$-pairs are indicated in the lists above by underlining j.

Suppose that the lemma holds for $n-1$; we will show that it holds for n. There are two cases to consider, depending upon whether $x_n = n$ or not. Figure 2.1 gives a example of how to find Hamilton path in R_4.

[Case I] If $x_n = n$ then we form a list of all $(n-1)!$ permutations with an n in the final position. The list starts at $12\cdots n$ and ends at $2x_2x_3\cdots x_n$.

Case I. 1234 → 2314.

1234	**213$\underline{4}$**	$\underline{4}$213	2413	1243	4123
1423	214$\underline{3}$	**$\underline{3}$214**	4321	3421	2341
4231	2431	324$\underline{1}$	**$\underline{1}$324**	**3124**	4312
3412	1342	4132	1432	314$\underline{2}$	**$\underline{2}$314**

Case II. 1234 → 2341.

1234	2134	3214	1324	3124	231$\underline{4}$
$\underline{4}$231	**243$\underline{1}$**	$\underline{1}$243	2143	4213	1423
4123	241$\underline{3}$	**$\underline{3}$241**	**4321**	**3421**	1342
3142	4312	1432	4132	341$\underline{2}$	**$\underline{2}$341**

(Case I). The permutations with bold font are obtained by putting 4 in position 4 in the permutations that are in the Hamilton path 123 → 231 in R_3. The permutations between the bold font permutations are blocks with i $(i = 1, 2, 3)$ in position 4. The underlined elements are the j's in the $(j, 4)$-pairs $(j = 1, 2, 3, 4)$. (Case II). The permutations in the first line are obtained by putting 4 in position 4 in the permutations that are in the Hamilton path 123 → 231 in $R(3)$. The permutations with the bold font are permutations in the Hamilton path 423 → 234 in R_3 (relabel the graph) with 1 in position 4. The permutations between the bold font permutations are blocks with 2, 3 in position 4. The underlined elements are the j's in the $(j, 4)$-pairs $(j = 1, 2, 3, 4)$.

Figure 2.2: Constructing Hamilton paths in R_4.

Such a list exists by the inductive assumption. Furthermore, the list has $(j, n-1)$-pairs for each $j = 1, 2, \cdots, n-1$. Let π, π' be a $(j, n-1)$-pair where, say $\pi = y_2 \cdots y_{n-1} j n$ and so $\pi' = j y_2 \cdots y_{n-1} n$. The arc $\pi \rightarrow \pi'$ is replaced with the following list.

$$\begin{array}{lcll} \pi & = & y_2 \cdots y_{n-1} j n & r_n \\ \tau & = & n y_2 \cdots y_{n-1} j & \\ & & \vdots \; \vdots & \\ \tau' & = & y_2 \cdots y_{n-1} n j & r_n \\ \pi' & = & j y_2 \cdots y_{n-1} n & \end{array}$$

The list from τ to τ' contains all $(n-1)!$ permutations that end in j. It exists by the inductive assumption. We do the above replacement for each $j = 1, 2, \cdots, n-1$ and thereby obtain a Hamilton path H in R_n from $12 \cdots n$ to $2 x_2 \cdots x_n$. Note that π,τ form a (n, n)-pair and that τ',π' form a (j, n)-pair, and thus H has a (j, n)-pair for each $j = 1, 2, \cdots, n$.

[Case II] If $x_n \neq n$, then there is some $1 \leq i < n$ such that $x_i = n$. Inductively, there are lists of all permutations ending with n or x_n that can be joined into a single path of length $2(n-1)!$ in R_n as shown below.

$$\begin{array}{lcll} \alpha & = & 12 \cdots (n-1) n & \\ & & \vdots \; \vdots & \\ \alpha' & = & 2 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n n & r_n \\ \beta & = & n 2 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n & \\ & & \vdots \; \vdots & \\ \beta' & = & 2 x_2 x_3 \cdots x_n & \end{array}$$

Note that the list above contains an (n, n)-pair. Lists of permutations ending with an element in the set $[n] \setminus \{n, x_n\}$ can be spliced in as they were in Case I; we do the splicing in the sublist starting at β and ending at β'. Each such splicing, say of those permutations ending x_k, creates both a (x_n, n)-pair and a (x_k, n)-pair. Thus we obtain a (j, n)-pair for each $j = 1, 2, \cdots, n$. □

Theorem 6 *For $n \geq 2$, there is a Hamilton path in R_n starting at $12 \cdots n$ and ending at $x_1 x_2 \cdots x_n$ if and only if $x_1 = 2$.*

Proof: This theorem follows from Lemmas 1 and 2, together with the observation that 12, 21 is the only Hamilton path in G_2. □

Theorem 7 *For $n \geq 2$, there is a Hamilton path in R_n starting at $x_1 x_2 \cdots x_n$ and ending at $y_1 y_2 \cdots y_n$ if and only if $x_2 = y_1$.*

Proof: This theorem follows from Theorem 6 and Theorem 5 in Chapter 1. □

By setting the ending vertex be any vertex that adjacent to the starting vertex in the previous theorem, we have the following corollary.

Corollary 1 *The digraph R_n is Hamiltonian for all $n \geq 3$.*

2.2 Hamilton paths and cycles in L_n

In the previous section, we study S_n as generated by right rotations. Similarly, we could consider the Cayley digraph $L_n = Cay(\{l_2, l_3, \cdots, l_n\} : S_n)$ where the generators are the left rotations $l_i = (1\ 2\ \cdots\ i-1\ i)$ $(2 \leq i \leq n)$. L_n can be obtained by reversing the directions of arcs in R_n. So if there is

a Hamilton path starting at $12\cdots n$ and ending at $2x_2x_3\cdots x_n$ in R_n, there is a Hamilton path in L_n starting at $2x_2x_3\cdots x_n$ and ending at $12\cdots n$. We can obtain the parallel theorems stated below.

Theorem 8 *There is a Hamilton path in L_n starting at $12\cdots n$ and ending at $x_1x_2\cdots x_n$ if and only if $x_2 = 1$.*

Theorem 9 *For $n \geq 2$, there is a Hamilton path (cycle) in L_n starting at $x_1x_2\cdots x_n$ and ending at $y_1y_2\cdots y_n$ if and only if $x_1 = y_2$.*

The proof of the two theorems is similar to the proof for the case R_n.

Corollary 2 *The digraph L_n is Hamiltonian for all $n \geq 2$.*

We have shown that R_n and L_n are Hamiltonian by using a **self-resemblance** technique. The self-resemblance technique takes advantage of self-resemblance phenomena, which is, in enumerating combinatorial objects, there are local structures resembling the whole structure. The defined (j, n)-pairs are such local structures in this problem, which we can insert all permutations with element j in position n between this pair. It is very interesting to notice that after the insertion, it produces a new $(j, n+1)$-pair, which makes insertion possible when n changes to $n+1$. That is, the self-resemblance attribute can be inherited.

Chapter 3

General Faber-Moore Digraphs

In this chapter, we prove that general Faber-Moore digraphs are Hamiltonian. The graph $G(4,2)$ is illustrated in Figure 3.2. We denote the generator $(i\ i-1\cdots 2\ 1)$ by g_i ($2 \leq i \leq k$). We denote the generator

$$x_1x_2\cdots x_k \rightarrow mx_1x_2\cdots x_{k-1}$$

by $g_{k+1}(m)$ $(m \in [n] \setminus \{x_1,\cdots,x_k\})$.

Recall in the previous chapter, the vertices in R_n are permutations of $[n]$. If we replace $[n]$ by set $C = \{a_1, a_2, \cdots, a_n\}$ and replace i by a_i $(i = 1, 2, \cdots, n)$, then the new graph $R_n(C)$ is isomorphic to R_n. We call a Hamilton path in $R_n(C)$ $H_n(C)$. Sometimes, we also call $H_n(C)$ **permutations of C**. According to Theorem 8 in Chapter 2, we have the following corollary.

Corollary 3 *There is a hamilton path in $R_n(C)$ starting at $a_1a_2\cdots a_n$ and ending at $x_1x_2\cdots x_n$ if and only if $x_1 = a_2$.*

There are

$$\frac{n!}{(n-k)!} = \left(\begin{array}{c} n \\ k \end{array} \right) k!$$

vertices in $G(n,k)$. The proof that $G(n,k)$ is Hamiltonian is based on a combinatorial explanation of the formula. We list all k-combinations from the set $[n]$ in a special order L, then for each k-combination C in L, replace C by a list of permutations of C.

There are two sections in this chapter. In the first section, we study k-combinations from $[n]$. In the second section, we study the Hamiltonicity in $G(n,k)$.

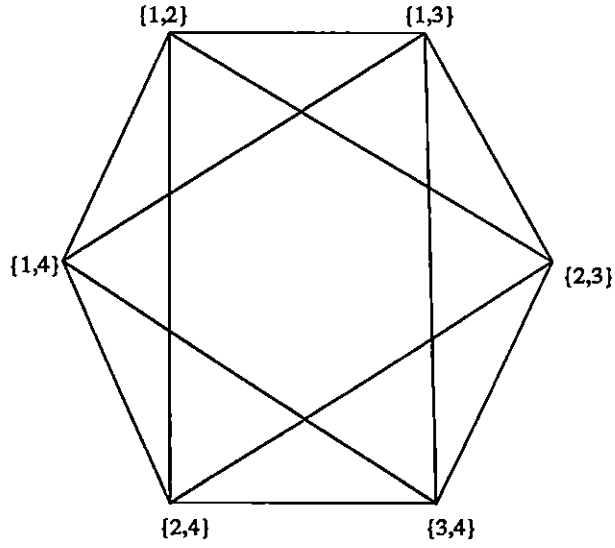


Figure 3.1: The combination graph $GC(4,2)$.

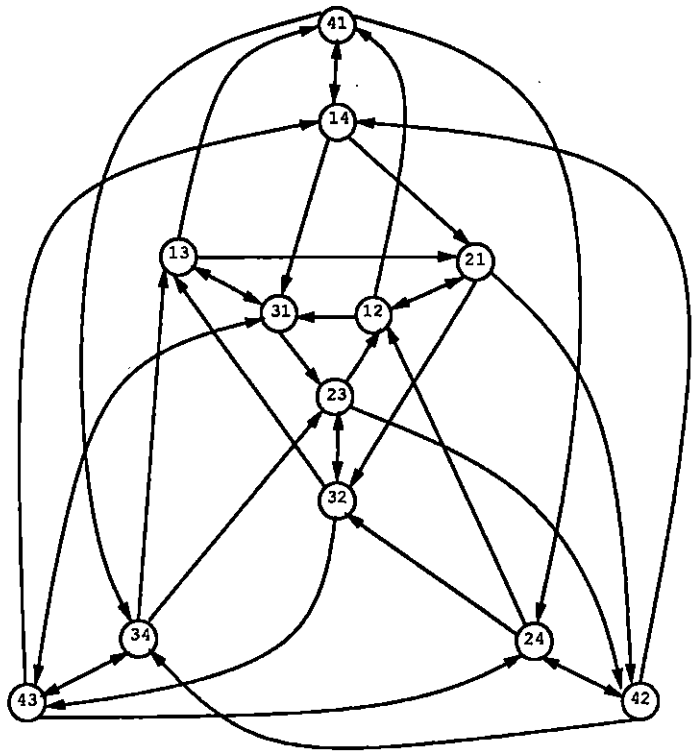


Figure 3.2: The digraph $G(4,2)$.

3.1 Hamilton-connectedness of combination graphs

Definition: Given a set $[n]$, a k-combination is a subset of $[n]$ with k elements. **k-combination graph** $GC(n,k)$ $(n \geq 2, n > k > 0)$ is a graph with vertex set $\{v \mid v \text{ is a } k\text{-combination of set } [n]\}$. Two vertices are adjacent if and only if the intersection of their corresponding k-combinations is a set of $k-1$ elements. We denote the order of $GC(n,k)$ by

$$C(n,k) = \begin{pmatrix} n \\ k \end{pmatrix}.$$

Figure 3.1 is the combination graph $GC(4,2)$.

There are many methods for listing k-combinations. For details see [ReNiDe] and [NiWi]. In this section, we prove that combination graphs are Hamilton-connected. First we give an easy lemma.

Lemma 3 *K_n is Hamilton-connected.*

The proof follows from the fact that any permutation of vertices is a Hamilton path in K_n.

Theorem 10 *$GC(n,k)$ ($n \geq 2$ and $n > k > 0$) is Hamilton-connected.*

Proof: We prove the theorem by induction on n. For each fixed $n \geq 2$, we prove that, given a pair of vertices u, v in $GC(n,k)$ $(0 < k < n)$, there is a Hamilton path starting at u and ending at v.

For $n = 2, 3$, $GC(n,k)$ is K_2, K_3. The theorem holds according to the previous lemma. Assume the theorem holds for $GC(m,l)$ where $0 < l < m <$

n and $l < k$. When $n = m + 1$, we partition the vertices of $GC(m+1, k)$ into two sets V_1, V_2, where V_1 is the set of all vertices that do not contain the element $m + 1$, and V_2 is the set of remaining vertices. Let G_1, G_2 be the subgraph of $GC(m+1, k)$ induced by V_1, V_2, respectively. We prove the theorem holds for $n = m+1$ according to three cases. We consider $1 < k < m$ in the following proof since if $k = m$ or $k = 1$, $GC(m+1, k)$ is K_{m+1}. The theorem holds according to Lemma 3.

[Case I] Both u and v are in G_1.

According to the assumption, there's a Hamilton path H_1 in G_1 starting at u and ending at v. Let α_1, α_2 be two successive vertices in the Hamilton path. Let

$$\alpha_1 = \{a_1, a_2, \cdots, a_{k-1}, a_k\}$$

and

$$\alpha_2 = \{a_1, a_2, \cdots, a_{k-1}, a_{k+1}\}.$$

Also, there is a Hamilton path H_2 in G_2 starting at

$$\beta_1 = \{a_1, a_2, \cdots, a_{k-2}, m+1, a_k\}$$

and ending at

$$\beta_2 = \{a_1, a_2, \cdots, a_{k-2}, m+1, a_{k+1}\}.$$

Since α_1 and β_1, α_2 and β_2 are adjacent in $GC(m+1, k)$, we can insert H_2 between α_1 and α_2. This is the Hamilton path for $GC(m+1, k)$.

[Case II] Both u and v are in G_2.

The proof is similar to Case I, we omit the proof here.

[Case III] One of u, v is in G_1 and the other in G_2.

Without loss of generality, we can assume u in G_1 and v in G_2. Let $\gamma(\gamma \neq u)$ be a k-combination in $[m]$. According to the inductive assumption, there is a Hamilton path H_1 in G_1 starting at u and ending at γ. Also, there is a Hamilton path H_2 in G_2 starting at $\gamma' = \gamma \setminus \{i\} \cup \{m+1\}$ and ending at v (choosing i such that $\gamma' \neq v$). Since γ and γ' are adjacent in $GC(m+1,k)$, H_1 followed by H_2 is the Hamilton path required. □

3.2 Hamilton paths and cycles in $G(n,k)$

A sufficient condition for the existence of a Hamilton path and cycle in $G(n,k)$ is described by the lemma below.

Lemma 4 *Let $n \geq k \geq 3$ and $x_2x_3\cdots x_k$ be a permutation of any $(k-1)$-combination C of $[n] \setminus \{2\}$. There is a Hamilton path H in $G(n,k)$ starting at $12\cdots k$ and ending at $2x_2x_3\cdots x_k$.*

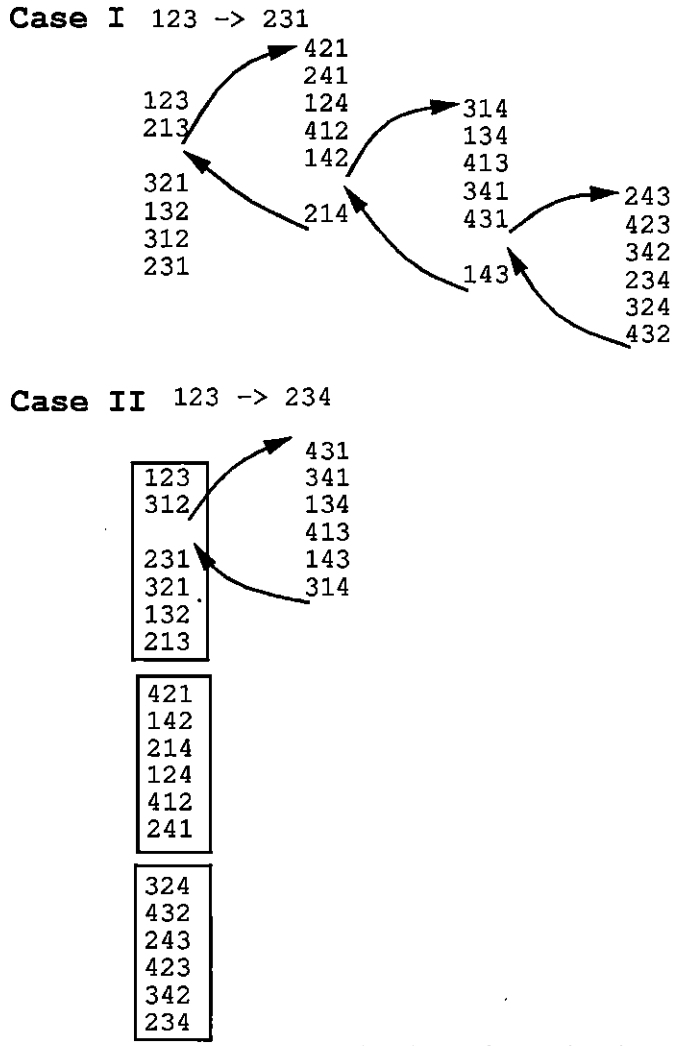
Proof: According to Lemma 2 in Chapter 2, the lemma holds if $n = k \geq 3$. If $n > k$, there are two cases to consider, depending on whether all $x_i \in [k](i = 2,\cdots,k)$ or not. Figure 3.3 gives an example of how to construct Hamilton paths in $G(4,3)$.

[Case I] If all $x_i \in [k](i = 2,\cdots,k)$, then according to Lemma 2, there is a Hamilton path H_k in R_k starting at $12\cdots k$ and ending at $2x_2x_3\cdots x_k$. Also, according to Theorem 10 there is a Hamilton path H in $GC(n,k)$ starting at $\{1,2,\cdots,k\}$. Let

$$C_i = \{a_1, a_2, \cdots, a_{k-1}, a_k\},$$

and

$$C_{i+1} = \{a_1, a_2, \cdots, a_{k-1}, a_{k+1}\}$$



(Case I). The Hamilton path in $GC(4,3)$ is $\{1,2,3\}$, $\{1,2,4\}$, $\{1,3,4\}$, $\{2,3,4\}$. Inserting permutations of one combination into the permutations of the followed permutations of combination. (Case II). The 3 boxes contain the permutations of combinations that contain 2, that is, $\{1,2,3\}, \{1,2,4\}, \{1,3,4\}$. The permutations of $\{1,3,4\}$ are inserted.

Figure 3.3: Constructing Hamilton Paths in $G(4,3)$.

be two successive vertices in H. Also let π, π' be a (a_k, k)-pair in $H_k(C_i)$ in $R_k(C_i)$ and let $\pi = a_1 a_2 \cdots a_{k-1} a_k$, $\pi' = a_k a_1 a_2 \cdots a_{k-1}$ (note : the elements of C_i and C_{i+1} may have to be re-indexed, but this may be done without loss of generality). According to Lemma 2, there exists $H_k(C_{i+1})$ in $R_k(C_{i+1})$ starting at $\tau = a_{k+1} a_1 a_2 \cdots a_{k-1}$ and ending at $\tau' = a_1 a_2 \cdots a_{k-1} a_{k+1}$. Since there are arcs from π to τ and from π' to τ', we can splice arc $\pi \rightarrow \pi'$ by inserting permutations of C_{i+1} between them and obtain the list below.

$$\begin{array}{lcll} \pi & = & a_1 a_2 \cdots a_{k-1} a_k & g_{k+1}(a_{k+1}) \\ \tau & = & a_{k+1} a_1 \cdots a_{k-2} a_{k-1} & \\ & & \vdots \; \vdots & \\ \tau' & = & a_1 a_2 \cdots a_{k-1} a_{k+1} & g_{k+1}(a_k) \\ \pi' & = & a_k a_1 \cdots a_{k-2} a_{k-1} & \end{array}$$

If we apply the above procedure to all the vertices (k-combinations) in H, then we have a Hamilton path in $G(n, k)$ starting at $12 \cdots k$ and ending at $2 x_2 x_3 \cdots x_k$.

[Case II] There is at least one $x_i \notin [k]$.

In graph $GC(n, k)$, we denote the subgraph induced by vertices (combinations) that contains 2 by G' and the subgraph induced by the other vertices to be G''. Obviously, G' is isomorphic to $GC(n-1, k-1)$ and G'' isomorphic to $GC(n-1, k)$. According to Theorem 10, there is a Hamilton path H in G'. Let C_i, C_{i+1}, C_{i+2} be three successive vertices in H, where

$$C_i = \{a_1, 2, a_3, \cdots, a_{k-2}, x, z\},$$

$$C_{i+1} = \{a_1, 2, a_3, \cdots, a_{k-2}, y, z\},$$

$$C_{i+2} = \{a_1, 2, a_3, \cdots, a_{k-2}, y, u\}.$$

We can sequentially arrange the permutations of C_i, C_{i+1}, C_{i+2} as shown below (again, the elements of C_i, C_{i+1} and C_{i+2} may have to be re-indexed).

$$\begin{array}{rcll}
\alpha & = & a_1 2 a_3 a_4 \cdots a_{k-2} x z & \\
 & & \vdots\ \vdots & \\
\alpha' & = & 2 a_1 a_3 a_4 \cdots a_{k-2} z x & g_{k+1}(y) \\
\beta & = & y 2 a_1 a_3 \cdots a_{k-2} z & \\
 & & \vdots\ \vdots & \\
\beta' & = & 2 y a_1 a_3 \cdots a_{k-2} z & g_{k+1}(u) \\
\gamma & = & u 2 y a_1 a_3 \cdots a_{k-2} & \\
 & & \vdots\ \vdots & \\
\gamma' & = & 2XXX \cdots XXX &
\end{array}$$

Since C_i has element x and C_{i+1} has element y, and the other parts are the same, we move x to the kth position in α', so by applying $g_{k+1}(y)$, we get to β, which is a permutation of C_{i+1}. The β' is obtained in the same way, moving element z to the kth position in β'. The X's in γ' are decided by the vertex following C_{i+2} in H. By applying the above procedure to all the vertices in G', we have a path in $G(n,k)$ starting at $12\cdots k$ and ending at $2x_2x_3\cdots x_k$ with $C(n-1,k-1)k!$ vertices. Also there is a Hamilton path H' starting at $\{1,3,4,\cdots,k+1\}$ in G''. Since $\{1,2,\cdots,k\}$ is adjacent to $\{1,3,4,\cdots,k+1\}$ in $GC(n,k)$, we can insert permutations of all k-combinations in H' into permutations of $\{1,2,\cdots,k\}$, in the same way as described in Case I. □

Lemma 5 *For any $n \geq 2$ and $x \in [n] \setminus \{2\}$, there is a Hamilton path in $G(n,2)$ starting at 12 and ending at $2x$ that contains at least one pair of* **twins**, *which are successive vertices in the Hamilton path and of the form*

pq, qp.

Proof: We prove the result by induction on n. If $n = 2$, the Hamilton path is 12 21. 12 and 21 are twins. Suppose the lemma holds for $n = k - 1$. If $n = k$, then for any $p \neq k$ the following is a path P in $G(k, 2)$.

$$k\,p,\ (p+1)\,k,\ k\,(p+1),\ \cdots,\ (k-1)\,k,\ k\,(k-1),\ 1\,k,\ k\,1,\ \cdots,\ p\,k.$$

We note that all the 2-combinations that contain element k are in P. If $x \in [k-1]$, according to the assumption for $n = k - 1$, there is a Hamilton path in $G(k-1, 2)$ starting at 12 and ending at $2x$ which contains at least one pair of twins, pq and qp. We can insert the path P between them to get the Hamilton path in $G(k, 2)$. If $x = k$, there is a Hamilton path H in $G(k-1, 2)$ starting at 12 and ending at $2\,(k-1)$. Let $p = 2$. Then P followed by H is the Hamilton path required. □

Theorem 11 *For $n \geq 2$, there is a Hamilton path in $G(n, k)$ starting at $12\cdots k$ and ending at $x_1x_2\cdots x_k$ if and only if $x_1 = 2$.*

Proof: This theorem follows from Lemmas 1, 4 and 5. □

Theorem 12 *For $n \geq 2$, there is a Hamilton path in $G(n, k)$ starting at $x_1x_2\cdots x_k$ and ending at $y_1y_2\cdots y_k$ if and only if $x_2 = y_1$.*

Proof: This theorem follows from previous theorem and the fact that $G(n, k)$ is vertex-symmetric [FaMo]. □

By setting the ending vertex be any vertex that adjacent to the starting vertex in the previous theorem, we have the following corollary.

Corollary 4 *The digraph $G(n, k)$ is Hamiltonian for all $n \geq 2$ and $n \geq k$.*

Chapter 4

Generating Permutations by a Tree of Transpositions

Permutation generation has a long and distinguished history as documented in Sedgewick's survey [Se]. It was actually one of the first nontrivial nonnumeric problems to be attacked by computers [Se]. Permutation generation algorithms can be used to test conjectures and theorems about many combinatorial objects. In some cases, conjectures might be suggested by the resulting data.

Among the permutation generation methods, generating permutations such that successive permutations are of some minimal difference, for example, that they differ by a transposition, are particularly useful. These algorithms are efficient in the sense that a small change is needed to obtain a new permutation from the previous one. The following well-known theorem explains when S_n can be generated by transpositions.

Theorem 13 *Let $G(V,E)$ be a graph with $V = \{1,2,\cdots,n\}$. If an edge $uv \in E$ is seen as a transposition (uv), $Cay(E : S_n)$ is connected if and only if G is connected.*

A proof can be found in Berge's book [Be].

Generating permutations by a tree of transpositions is equivalent to finding a Hamilton path in the corresponding tree Cayley graph. If G is a path labelled with $1, 2, \cdots, n$, then a generating method is described by the well-known Johnson-Trotter algorithm [Jo][Tr]. The algorithm is obtained by a combinatorial explanation of $n! = n\,(n-1)!$. For each of the $(n-1)!$ permutations of $[n-1]$, the element n may be inserted in n positions. Figure 4.1 illustrates how this algorithm works for $n = 4$. In order to generate permutations of [4], we first generate all the permutations of [3], then for each of the generated permutations, we put element 4 in four different positions. As we see from the figure, element 4 moves from left to right, then from right to left. This is repeated until all the permutations of [3] are covered.

A consequence of Theorem 13 is that S_n can be generated by a tree of transpositions, whose vertices are labeled with $1, 2, \cdots, n$. Also if any edge is deleted from the tree, S_n can not be generated. We develop an algorithm to generate permutations by a tree of transpositions. Our algorithm is based on the proof that there is a Hamilton path in $Cay(T : S_n)$ [Sl]. Proofs for the case where T is a star can be found in Nigam, Sahni and Krishnamuthy [NiSaKr], Jwo, Lakshmivarahan and Dahll [JwLaDh], Akl, Duprat and Ferreira [AkDuFe]. We first introduce a slight modification of Slater's proof, then we show how to convert the proof to an efficient algorithm.

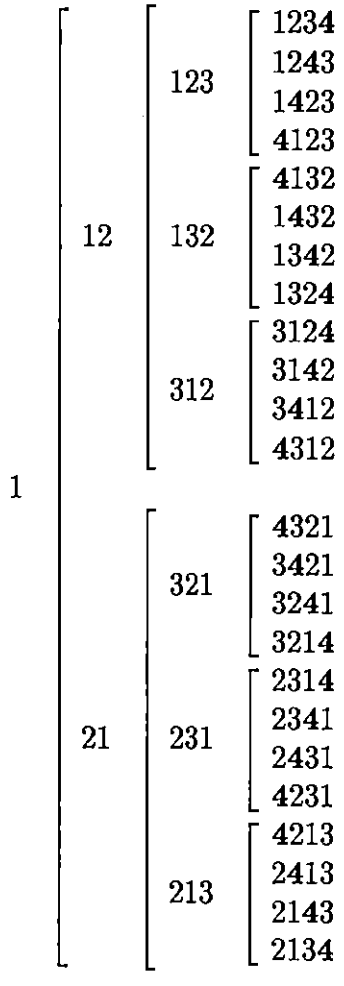


Figure 4.1: An example of the Johnson-Trotter algorithm.

4.1 Slater's proof

Let $\pi(i)$ be the i-th element in the permutation π. Then $\pi^{-1}(i)$ denotes the position of element i in π. For example, if

$$\pi = \begin{pmatrix} 1\,2\,3\,4\,5 \\ 3\,5\,4\,1\,2 \end{pmatrix},$$

then

$$\pi^{-1} = \begin{pmatrix} 1\,2\,3\,4\,5 \\ 4\,5\,1\,3\,2 \end{pmatrix}.$$

Theorem 14 *Let T be a tree of transpositions with $n \geq 4$ vertices and s and d be integers, where $1 \leq s, d \leq n$. Starting at an arbitrary permutation π_1, the graph $Cay(T : S_n)$ has a Hamilton path $\pi_1, \pi_2, \cdots, \pi_{n!}$ such that $\pi_{n!}(d) = \pi_1(s)$.*

Proof: Since Cayley graphs are vertex-symmetric, it is sufficient to prove the theorem for $\pi_1 = 12\cdots n$. The proof is by induction on n. There are two non-isomorphic trees when $n = 4$. Figures 4.2 and 4.4 are the corresponding tree Cayley graphs. In each of the two Cayley graphs, for two integers s and d, where $1 \leq s, d \leq 4$, there is a Hamilton path starting at 1234 and satisfying $\pi_{4!}(d) = s$. The related paths are listed in Figure 4.3 and Figure 4.5. If $n > 4$, let (p, q) be an edge of T, where p is a leaf. We may assume $p \neq d$ since T has at least two leaves. Let $L(T_n, d, s)$ denote a list of $n!$ permutations such that successive permutations differ by a transposition of an edge in T_n and $\pi_{n!}(d) = \pi_1(s)$. Let $i_1, i_2, \cdots, i_n$, where $\pi_1(p) = i_1$, be a sequence of of vertices in tree T where $i_n \neq s'$ ($s' = \pi_1(s)$). Let T'_{n-1} be the

tree obtained by removing vertex p from T_n. Then the following is a required list:

$$
\begin{array}{l}
L(T'_{n-1},\ q,\ \pi_1^{-1}(i_2)) \\
Swap(p,q) \\
L(T'_{n-1},\ q,\ \pi_{(n-1)!+1}^{-1}(i_3)) \\
Swap(p,q) \\
\quad \vdots\ \vdots \\
L(T'_{n-1},\ q,\ \pi_{(n-2)(n-1)!+1}^{-1}(i_n)) \\
Swap(p,q) \\
L(T'_{n-1},\ d,\ \pi_{(n-1)(n-1)!+1}^{-1}(s'))
\end{array}
$$

$L(T'_{n-1}, q, \pi_{j(n-1)!+1}^{-1}(i_m))(2 \leq m \leq n)$ is a list of $(n-1)!$ permutations such that every permutation has element i_{m-1} in position p. The last list $L(T'_{n-1}, d, \pi_{(n-1)(n-1)!+1}^{-1}(s'))$ is a list of $(n-1)!$ permutations such that every permutation has element i_n in position p. For two successive lists L_1, L_2 in the above listing, the first permutation in L_2 is obtained by applying $Swap(p,q)$, the transposition (pq), to the last permutation in L_1. □

The proof of Slater is slightly different in that his final conclusion is $\pi_{n!}(d) = s$, rather than $\pi_{n!}(d) = \pi_1(s)$. Our proof helps lead to an efficient algorithm.

4.2 The algorithm

One way of representing a path is by listing all vertices. For example, we can represent a Hamilton path in Cayley graph where T is a 3-path by

$$123, 213, 231, 321, 312, 132.$$

In Cayley graphs, we can represent a path by listing the first vertex and then a sequence of generators. For example, the previous Hamilton path can be represented by

$$123, (12), (23), (12), (23), (12). \qquad (I)$$

The second method of representation can be changed easily to represent a variety of paths with the same structure. For example,

$$12345, 14325, 14352, 15342, 15324, 12354.$$

can be represented as

$$12345, (24), (45), (24), (45), (24). \qquad (II)$$

Observe that the transpositions in (II) can be obtained from the transpositions in (I) by relabeling the transpositions $\{(12), (23)\}$ by $\{(24), (45)\}$. In our algorithm, we use the second method of representation. Hamilton paths in Figures 4.3 and 4.5 should be viewed as sequences of transpositions. The algorithm consists of two passes. In the first pass, the "top-down" pass, we keep on deleting leaves of T until we reach the base case, $n = 4$, by using the method discussed in the proof. The tree of 4 vertices can be either a path or a star. But the labels of the tree are different, depending on structure and labels of T. We can change transpositions in Figures 4.3 and 4.5 to suitable transpositions by relabeling them, as discussed above. In the second pass, the "bottom-up" pass, we get the transpositions in the higher case ($n = k + 1$), by applying the known transpositions in the lower case ($n = k$). The two passes can be described by recursions, as in Figure 4.7. Pascal code for this

algorithm is listed in the appendix.

In the algorithm, procedure *FindDeleteLeaf(d)* is used to find an edge (p, q), where p is a leaf and $p \neq d$. It uses constant time. Function *Next* and variable *dir* are used to choose $i_1, i_2, \cdots, i_n$, as is described in the proof. *Next* returns the first unused element found along the *dir* direction. We use $dir = +1$ to represent circularly searching from left to right and $dir = -1$ to represent the opposite direction. For details see appendix. Procedure *Swap(p, q)* exchanges elements in position p and q. It uses constant time. Figure 4.6 shows the output of the algorithm when $n = 5$.

The algorithm uses $O(n)$ space to store the tree T and the initial permutation π, which is $12 \cdots n$. We use $f(n)$ to denote the time complexity of the algorithm when T has n vertices. From the algorithm, we have the following recursive relation:

$$f(n) = n \cdot f(n-1) + c_1 \cdot n + c_2,$$

where c_1, c_2 are constants. Constant c_2 comes from the preprocessing before the *for* loop. The term $c_1 \cdot n$ comes from the operations inside the loop. Note though we do not know whether *Next* uses constant time or not in each iteration, it uses $O(n)$ time in the $(n-1)$ iterations because the circular search goes through the entire list once during $(n-1)$ iterations and the length of the list is n. Letting $c = c_1 + c_2$ and $g(n) = f(n) + c_1$, we have

$$g(n) = n \cdot g(n-1) + c.$$

That is, $g(n) \in O(n!)$, and thus $f(n) \in O(n!)$. If each permutation is output, then the running time is $O(n \cdot n!)$ because $O(n)$ time is needed to output a

permutation. If the output is a sequence of generators, the running time is $O(n!)$ because output a transposition can be done in constant time.

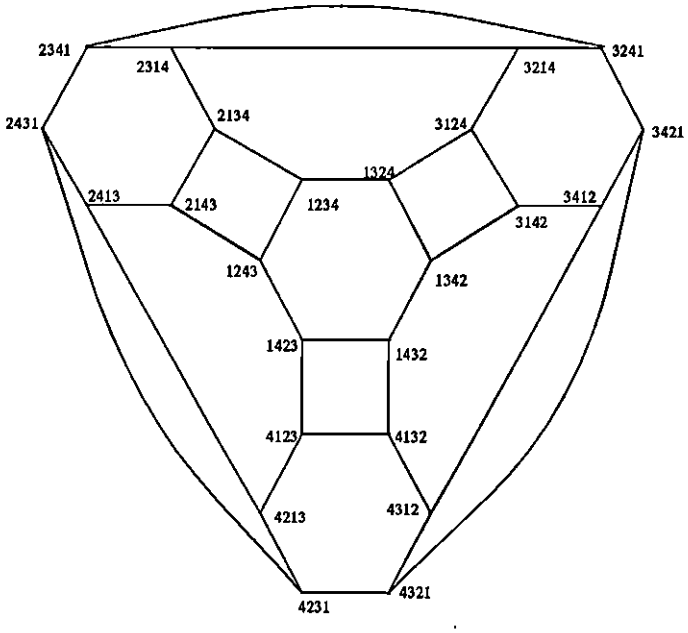


Figure 4.2: The 4-path graph : $Cay(\{(12),(23),(34)\} : S_4)$

Path 1 :	1234	1324	1342	1432	1423	4123	4132	4312
	4321	4231	4213	2413	2431	2341	3241	3421
	3412	3142	3124	3214	2314	2134	2143	1243
Path 2 :	1234	1243	1423	1432	1342	1324	3124	3142
	3412	3421	3241	3214	2314	2341	2431	4231
	4321	4312	4132	4123	4213	2413	2143	2134
Path 3 :	1234	1243	1423	1432	1342	1324	3124	3142
	3412	4312	4132	4123	4213	2413	2143	2134
	2314	3214	3241	2341	2431	4231	4321	3421
Path 4 :	1234	1243	1423	1432	1342	1324	3124	3142
	3412	3421	4321	4231	2431	2341	3241	3214
	2314	2134	2143	2413	4213	4123	4132	4312

Figure 4.3: Hamilton paths in the 4-path graph.

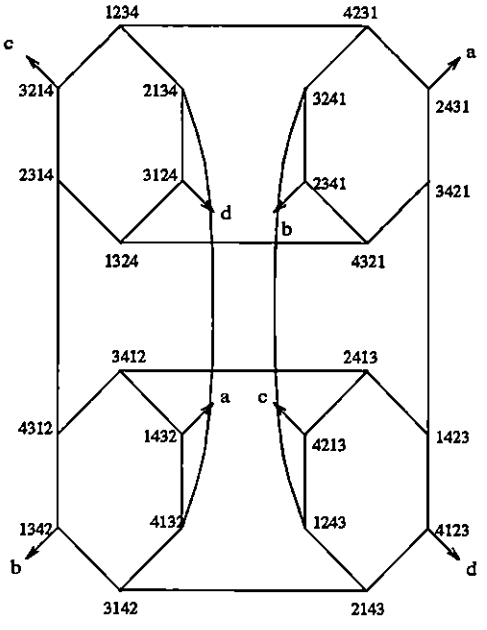


Figure 4.4: The 4-star graph : $Cay(\{(12),(13),(14)\} : S_4)$.

Path 1 :	1234	4231	3241	1243	4213	3214	2314	4312
	1342	2341	4321	3421	2431	1432	3412	2413
	1423	4123	2143	3142	4132	2134	3124	1324
Path 2 :	1234	4231	3241	1243	4213	3214	2314	4312
	1342	2341	4321	1324	3124	2134	4132	3142
	2143	4123	1423	3421	2431	1432	3412	2413
Path 3 :	1234	4231	3241	1243	4213	3214	2314	1324
	4321	2341	1342	4312	3412	2413	1423	3421
	2431	1432	4132	2134	3124	4123	2143	3142
Path 4 :	1234	3214	4213	1243	3241	2341	4321	1324
	2314	4312	1342	3142	2143	4123	3124	2134
	4132	1432	3412	2413	1423	3421	2431	4231

Figure 4.5: Hamilton paths in the 4-star graph.

12345	12543	15243	15342	13542	13245	31245	31542
35142	35241	32541	32145	23145	23541	25341	52341
53241	53142	51342	51243	52143	25143	21543	21345
21435	21534	25134	25431	24531	24135	42135	42531
45231	45132	54132	51432	15432	14532	41532	41235
14235	12435	12534	15234	51234	52134	52431	54231
54321	53421	53124	51324	51423	15423	15324	13524
13425	14325	14523	41523	41325	43125	34125	31425
31524	35124	35421	34521	43521	45321	45123	54123
54213	54312	53412	53214	52314	52413	25413	25314
23514	23415	32415	34215	43215	42315	24315	24513
42513	45213	45312	43512	34512	35412	35214	32514
32154	32451	34251	34152	31452	31254	13254	13452
14352	14253	12453	12354	21354	21453	24153	42153
41253	41352	43152	43251	42351	24351	23451	23154

Figure 4.6: Generating 5! when $T = \{(1,2),(2,3),(3,4),(3,5)\}$ with $d = 5, s = 4$.

```
procedure L(T : tree; d, s : 1..n);
local p, q, ps;
begin
    if |T| = 4 then BaseCase(T, d, s)
    else begin
        ps ← π(s);
        (p, q) ← FindDeleteLeaf(d);
        if ps =Next(π(p), +1) then dir ← +1 else dir ← −1;
        i ← π(p);
        for n − 1 iterations do begin
            i ← Next(i, dir);
            L(T − {p}, q, π^{-1}(i));
            Swap( p, q );
        end;
        L(T − {p}, d, π^{-1}(ps));
    end;
end { of L };
```

Figure 4.7: Pseudo-code algorithm.

Chapter 5

Conclusions and Open Problems

In this thesis, we devote most of our efforts to studying Hamiltonicity in Cayley graphs (digraphs) that related to a given group with given generators. These kinds of Cayley graphs (digraphs) are usually have their origins in the study of possible applications in design of parallel architectures. Proof by induction is the basic method in attacking these problems. First, we find the Hamilton cycles (paths) in base cases, and check to see certain conditions (inductive conditions) hold so that the induction can proceed from lower cases to higher cases. The inductive conditions are different in different problems. In Chapter 2, existence of the defined (j, n) pair is such condition. In Chapter 4, the condition $\pi_{n!}(d) = \pi_1(s)$ is the inductive condition. Finding the inductive conditions is the most important part in the proof.

Proof by induction is effective in proof of Hamiltonian property in Cayley graphs with consistent generators, that is, the generators appearing in the lower cases will also appear in higher cases. If this condition is not satisfied, different approaches are required. In Sections 1 and 2 of this chapter, we describe two of these problems. In Section 3, we list more problems related

with algorithms on Cayley graphs.

5.1 Cayley graphs: extension of σ-τ problem

The σ-τ problem is to prove

$$Cay(\{(12), l_n, r_n\} : S_n)$$

where $l_n = (1\,2\,\cdots\,n)$, $r_n = (n\;n-1\;\cdots\;2\,1)$, is Hamiltonian. The generators of this Cayley graph makes it impossible to use proof by induction directly. In his Ph.D thesis, Compton [CoWi] solved this problem by transforming it to the problem of generating permutations by doubly adjacent transpositions, which can be proved by induction. Unfortunately, the proof is very complicated and a simple proof of this problem is still open [Ru].

We consider extending the σ-τ problem by replacing the transposition (12) by transposition (ij). However, subset $\{(ij), l_n, r_n\}$ may not generate S_n. For example, $Cay(\{(13), l_4, r_4\} : S_4)$ has three connected components:

$$1234, 2341, 3412, 4123, 2143, 1432, 4321, 3214;$$

$$2134, 1342, 3421, 4213, 1243, 2431, 4312, 3124;$$

and

$$1324, 3241, 2413, 4132, 3142, 1423, 4231, 2314.$$

The following theorem explains when the generator set generates S_n.

Theorem 15 *If $1 \leq i < j \leq n$ and $(j - i)$ is relatively prime to n, subset $\{(ij), l_n, r_n\}$ generates S_n.*

Proof: We try to generate as many transpositions as possible by the given

1	2	$\cdots$	$i-1$	i	$i+1$	$\cdots$	$j-1$	j	$j+1$	$\cdots$	$n-1$	n	l_n
2	3	$\cdots$	i	$i+1$	$i+2$	$\cdots$	j	$j+1$	$j+2$	$\cdots$	n	1	l_n
					$\vdots$		$\vdots$						$\vdots$
u	$u+1$	$\cdots$	$u+i-2$	$u+i-1$	$u+i$	$\cdots$	$u+j-2$	$u+j-1$	$u+j$	$\cdots$	$u-2$	$u-1$	(ij)
u	$u+1$	$\cdots$	$u+i-2$	$u+j-1$	$u+i$	$\cdots$	$u+j-2$	$u+i-1$	$u+j$	$\cdots$	$u-2$	$u-1$	r_n
$u-1$	u	$\cdots$	$u+i-3$	$u+j-2$	$u+j-1$	$\cdots$	$u+j-3$	$u+j-2$	$u+i-1$	$\cdots$	$u-3$	$u-2$	r_n
					$\vdots$		$\vdots$						$\vdots$
1	2	$\cdots$	$u+i-2$	$u+j-1$	$u+i$	$\cdots$	$u+j-2$	$u+i-1$	$u+j$	$\cdots$	$n-1$	n	

generator set. We obtain the transposition $(u+i-1\ u+j-1)$ by applying l_n $(u-1)$ times, then applying transposition (ij), and finally, applying r_n $(u-1)$ times, as described below.

1	2	$\cdots$	$i-1$	i	$i+1$	$\cdots$	$j-1$	j	$j+1$	$\cdots$	$n-1$	n	l_n
2	3	$\cdots$	i	$i+1$	$i+2$	$\cdots$	j	$j+1$	$j+2$	$\cdots$	n	1	l_n
					$\vdots$		$\vdots$						$\vdots$
u	$u+1$	$\cdots$	$u+i-2$	$u+i-1$	$u+i$	$\cdots$	$u+j-2$	$u+j-1$	$u+j$	$\cdots$	$u-2$	$u-1$	(ij)
u	$u+1$	$\cdots$	$u+i-2$	$u+j-1$	$u+i$	$\cdots$	$u+j-2$	$u+i-1$	$u+j$	$\cdots$	$u-2$	$u-1$	r_n
$u-1$	u	$\cdots$	$u+i-3$	$u+j-2$	$u+j-1$	$\cdots$	$u+j-3$	$u+j-2$	$u+i-1$	$\cdots$	$u-3$	$u-2$	r_n
					$\vdots$		$\vdots$						$\vdots$
1	2	$\cdots$	$u+i-2$	$u+j-1$	$u+i$	$\cdots$	$u+j-2$	$u+i-1$	$u+j$	$\cdots$	$n-1$	n	

Let $u = 1, 2, \cdots, n+1$, we have the transposition list below.

$$
\begin{array}{l}
(i,j) \\
(i+1,j+1) \\
\vdots\ \vdots \\
(n-j+i,n) \\
(n-j+i+1,1) \\
\vdots\ \vdots \\
(n,j-i) \\
(1,j-i+1) \\
\vdots\ \vdots \\
(i-1,j-1)
\end{array}
$$

Since the transpositions in the list can be obtained from one another, $Cay(\{(i,j), l_n, r_n\} : S_n)$ is connected if and only if $Cay(\{(1,j-i+1), l_n, r_n\} : S_n)$ is connected.

Let $t = j - i + 1$ and rewrite the transposition list as follow.

$$\begin{array}{l}(1,u)\\(2,u+1)\\\vdots\ \vdots\\(n-u+1,n)\\(n-u+2,1)\\\vdots\ \vdots\\(n,u-1)\end{array}$$

We construct a graph $G(V,E)$ with $V = \{1,2,\cdots,n\}$ and $pq \in E$ if and only if (pq) is a transposition appearing in the transposition list. Since G is 2-regular, G is a cycle or union of cycles. We prove if $(u-1)$ is relatively prime to n, then G is a cycle of length n. Let the length of the cycle be k. The cycle is

$$1, u, 2u-1, 3u-2, \cdots, (k-1)u-(k-2).$$

We have $k(u-1) = mn$, where m is a integer. Since $(u-1)/n = m/k$ and $(u-1)$ is relatively prime to n, $k \geq n$. Since $k \leq n$, $k = n$. Since the transposition graph is connected, $Cay(\{(i,j), l_n, r_n\} : S_n)$ is connected. □

By this theorem and the Conjecture 1 in Chapter 1, we have the following conjecture.

Conjecture 3 *If* $1 \leq i < j \leq n$ *and* $j-i$ *is relatively prime to* n, $Cay(\{(i,j), l_n, r_n\} : S_n)$ *is Hamiltonian.*

5.2 Cayley digraphs: variation of the Faber-Moore digraphs

In the σ-τ problem, upon deleting one of l_n, r_n, the resulting Cayley digraph is strongly connected. Whether the resulting digraph has a Hamilton path is still open. The same question can also be asked for the Cayley digraph obtained by deleting one of the l_n, r_n from the Cayley graphs discussed in the previous section.

In the Faber-Moore digraph $G(n,k)$, if $n \geq k+1 \geq 2r \geq 4$, we may remove $r-1$ generators $g_2, g_3, \cdots, g_r$ from the generator set to get a new digraph $G_r(n,k)$. We conjecture that $G_r(n,k)$ is Hamiltonian. In [CoFi], they prove that if $n \geq k \geq 2r \geq 4$, $G_r(n,k)$ has diameter $k+r-1$. The following is a Hamilton cycle in $G_2(4,3)$.

123	312	431	243	124	412
241	324	132	413	341	134
213	321	432	143	214	421
342	234	423	142	314	231

It is interesting to notice that we can not apply proof by induction directly in proving the Hamiltonicity of $G_r(n,k)$. The reason is that generators appearing in the lower cases disappear in higher cases. So the induction does not hold any longer.

Bibliography

[AkHaKr] S. B. Akers, D. Harel and B. Krishnamurthy, "The star graph: an attractive alternative to the n-cube", *International Parallel Processing Conference*, 1987, 393-400.

[AkKr] S. B. Akers, B. Krishnamurthy, "A Group Theoretic Model for Interconnection Networks", Proc. *International Conference on Parallel Processing*, 1986, 216-233.

[AkDuFe] S. G. Akl, J. Duprat and A. G. Ferreira, "Building Hamiltonian Circuits and Paths in Star Graphs", *Technical Report No. 90-22, Laboratoire de l'Informatique du Parallelisme, Ecole Normale Superieure de Lyon, Lyon, France*, 1990.

[AnBaRo] F. Annexstein, M. Baumslag, and A. L Rosenberg, "Group Action Graphs and Parallel Architectures", *SIAM J. COMPT.*, Vol. 19, No. 3, June 1990, 544-569.

[Be] C. Berge, *Principles of Combinatorics*, Academic Press, New York, 1971.

[BeDe] J.C. Bermond and C. Belorme, "Strategies for Interconnection Networks: Some Methods from Graph Theory", *J. of Parallel and Distributed Computing*, 3, 433-449 (1986).

[BoMu] J.A Bondy and U.S.R. Murty, *Graph Theory with Applications*, Elsevier North Holland Inc., New York, 1979.

[ChQu] C.C. Chen and N.F. Quimpo, "Onstrongly Hamiltonian abelian group graphs", *Combinatorial Mathematics VIII, Lecture Notes in Mathematics*, Vol. 884, Springer-Verlag, Berlin, 23-34 (1981).

[CoFi] F. Comellas and M. A. Fiol, "Vertex symmetric digraphs with small diameter", submitted.

[CoSlWi] J. H. Conway, N.J.A. Sloane and A. R. Wilks, "Gray Codes for Reflection Groups", *Graphs and Combinatorics* 5, 315-325 (1989).

[CoWi] R.C. Compton, S. G. Williamson, "Doubly adjacent Gray codes for the symmetric group", *Technical Report Number CS91-220*, Dept. of Computer Sci., Univ. of California, San Diego, 1991.

[De] M.J. Dinneen, "Algebra method in constructing networks", *Master's thesis, University of Victoria,* Victoria, Canada, 1991.

[FaMo] V. Faber and J. W. Moore, "High-Degree Low-Diameter Interconnection Networks With Vertex Symmetry: The Directed Case", *Technical Report, Los Alamos National Laboratory*, 1990.

[Fe] M. Fellows, "Problem Corner", *Contemporary Mathematics*, Volume 89, 1989.

[Fr] J. B. Fraleigh, *A First Course In Abstract Algebra*, third edition, Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.

[Ga] J.A. Gallian, *Contemporary Abstract Algebra*, D. C. Heath and Company, Toronto, 1986.

[GaJo] M.R. Gary and D.J. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco, 1979.

[Go] R. J. Gould, "Updating the Hamiltonian Problem- A Survey", *Journal of Graph Theory*, Vol. 15, No. 2, 1991, 121-157.

[Gor] D. M. Gordon, "Parallel Sorting on Cayley Graphs", *Algorithmica*, 6(1991), 554-561.

[Gow] C. GowriSankaran, "Broadcasting on Recursively Decomposable Cayley Graphs", *submitted.*

[Gr] R. P. Grivnaldi, *Discrete and Combinatorial Mathematics*, 2nd Edition, P455, Ex 4(a), Addison Wesley, New York, 1989.

[Ja] B. Jackson, "Hamiltonian cycles in regular 2-connected graphs", *Journal Combinatorial Theory* , B, 29(1980), 27-46.

[Jo] S. M. Johnson, "Generation of permutations by adjacent transpositions", *Math. Comp.*, 17(1963), 282-285.

[JwLaDh] J.S. Jwo, S. Lakshmivarahan, and S.K. Dhall, "Embedding of Cycles and Grids in Star Graphs", *Proc. Second IEEE Symp. Parallel and Distributed Processing*, Dallas, Texas, 1990, 540-547.

[KoLi] V. L. Kompel'makher and V. A. Liskovets, "Sequential generation of arrangements by means of a basis of transpositions," *Kibernetica*, 3(1975), 17-21.

[Ma] D. Marusic, "Hamiltonian circuits in Cayley graphs", *Discrete Mathematics,* 46(1983) 49-54.

[NiSaKr] M. Nigam, S. Sahni, and B. Krishnamurthy, "Embedding Hamiltonians and Hypercubes in Star Interconnection Graphs", *Proc. International Conference on Parallel Processing*, (1990) 1-8.

[NiWi] A. Nijenhuis and H.S. Wilf, *Combinatorial Algorithms*, Second Edition, Academic Press, New York, 1978.

[ReNiDe] E.M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

[Ru] F. Ruskey, " Problem Lists", *Private communication, 1990.*

[RuSa] F. Ruskey and C. Savage, "Hamilton Cycles which Extend Transposition Matchings in Cayley Graphs of S_n", to appear in *SIAM J. Disc. Math.*.

[Sa] C. D. Savage, "Generating Permutations with k-Differences", *SIAM J. Disc. Math.*, Vol 3, No. 4, 1990, 561-573.

[Se] R. Sedgewick, "Permutation generation methods", *Computing Surveys*, Vol. 9, No. 2, June 1977.

[Sl] P. J. Slater, "Generating all permutations by graphical transpositions", *Ars Combinatoria*, 5 (1978), 219-225.

[StWh] D. Stanton and D. White, *Constructive Combinatorics*, Springer-Verlag, New York, 1986.

[Tc] M. Tchuente, "Generation of permutations by graphical exchanges", *Ars Combinatoria*, 14 (1982), 115-122.

[Tr] H. F. Trotter, "PERM (Algorithm 115)", *Communications of the ACM*, 5, No. 8 (1962), 434-435.

[Wi] H.S. Wilf, *Private communication*, 1992.

[Wit] D.S. Witte, "On Hamiltonian circuits in Cayley digrams", *Discrete Mathematics*, 38 (1982), 99-108.

[WiGa] D. Witte and A. Gallian, "A survey: Hamiltonian cycles in Cayley graphs", *Discrete Mathematics*, 51 (1984), 293-304.

Appendix

In this appendix, we list the Pascal code for the generating of permutations by a tree of transpositions.

```
program GeneratePerm(input, output);
const
  MAXN=20;
  RIGHT=1;
  LEFT=-1;
  LINE=2; {2 leaves in a 4 node line}
  STAR=3; {3 leaves in a 4 node star}

type
  advertex=^lvertex;
  lvertex=record
    int:integer;
    next:advertex
  end;
  transposition=record
    pos1,pos2:integer;
   end;
   permutation=array[1..4] of integer;

var
  i,j,nn:integer;
  desposition,srcposition,count:integer;
  space : array[1..MAXN] of 1..MAXN;
                          {the permutation}
  positionof:array[1..MAXN] of 1..MAXN;
                         {inverse of space}
  hpath4:array[LINE..STAR,1..4,1..23]
                          of transposition;
  pathindex:array[1..4,1..4] of 1..4;
  edge:array[LINE..STAR,1..3] of transposition;
```

```
                {edges of line and star in n=4}
  usedelement:array[1..MAXN] of boolean;
                  {true if not in current tree}
  tree: array[1..MAXN] of advertex;
                     {adjacency list for tree}
  degree: array[1..MAXN] of integer;
                    {degrees of nodes in tree}
  head: advertex;                {list of leaves}
  NumLeaves : integer;         {number of leaves}

procedure  OutPerm;              {print out perm}
var t : integer;
begin
  count:=count+1;
  for t:=1 to  nn do  write(space[t]:1);
  write('   ');
  if count = 6 then
    begin writeln; count:=0; end;
end;

procedure Interchange(tr:transposition);
  var temp: integer;

begin
  positionof[space[tr.pos1]]:=tr.pos2;
  positionof[space[tr.pos2]]:=tr.pos1;
  temp:=space[tr.pos1];
  space[tr.pos1]:=space[tr.pos2];
  space[tr.pos2]:=temp;
end;

procedure  Deal4(despos,srcpos:integer);
                       {Base case processing}
  var tr1,tr: transposition;
      x: array[1..4] of integer;
```

```
      d,s,i,temp,pathid:integer;
      tpt : advertex;

begin
       OutPerm;
  case NumLeaves of
  LINE:
    begin
      x[1]:=head^.int;
      x[2]:=tree[x[1]]^.int;
      tpt:=head^.next;
      x[4]:=tpt^.int;
      x[3]:=tree[x[4]]^.int;
    end;
  STAR:
    begin
      temp:=head^.int;
      x[1]:=tree[temp]^.int;
      tpt:=tree[x[1]];
      x[2]:=tpt^.int;
      tpt:=tpt^.next;
      x[3]:=tpt^.int;
      tpt:=tpt^.next;
      x[4]:=tpt^.int
    end
  end{ case };

  for i:=1 to 4 do
  begin
    if x[i]=despos then d:=i;
    if x[i]=srcpos then s:=i;
  end;
  for i:=1 to 4 do
  if s=pathindex[i,d] then pathid:=i;
  for i:=1 to 23 do
```

```
    begin
      tr1:=hpath4[NumLeaves,pathid,i];
      tr.pos1:= x[tr1.pos1];
      tr.pos2:= x[tr1.pos2];
      Interchange(tr);
      OutPerm;
    end;
end{ of Deal4 };

procedure ListPerm(n,despos, srcpos:integer);
  var count,i,next, curno,curpos:integer;
      dpos,srcelement,fixelem,fixpos:integer;
      tr: transposition;
      direction:integer;
      leafstock:advertex;
      leafval:integer;
      adjofleaf:integer;

function FindDeleteLeaf(dpos:integer):integer;
begin
   NumLeaves:=NumLeaves-1;
   if head^.int = dpos then begin
      leafstock := head^.next;
      head^.next := leafstock^.next;
      end
   else
     begin
       leafstock:=head;
       head := head^.next;
     end;
   leafval := leafstock^.int;
   FindDeleteLeaf:=leafstock^.int;
end{ of FindDeleteLeaf };

procedure RecoverLeafList;
begin
```

```
  leafstock^.next:=head;
  head := leafstock;
  NumLeaves:=NumLeaves+1;
end{ RecoverLeafList };

procedure DeleteTreeLeaf(leaf:integer);
  var p,q: advertex;
      adjv:integer;

begin
  p:=tree[leaf];
  adjv:=p^.int;
  adjofleaf:=adjv;
  tree[leaf]:=nil;
  degree[leaf]:=0;
  p:=tree[adjv];
  if leaf=p^.int then
    begin
      tree[adjv]:=p^.next;
      dispose(p);
    end
  else
    begin
      while p^.int<>leaf do
      begin
   q:=p;
   p:=p^.next;
      end;
      q^.next:=p^.next;
      dispose(p);
  end {if};
  degree[adjv]:=degree[adjv]-1;
  if degree[adjv]=1 then
          {adjv is leaf now}
  begin
    NumLeaves:=NumLeaves+1;
```

```
    new(p);
    p^.int:=adjv;
    q:=head^.next;
    head^.next:=p;
    p^.next:=q;
  end;
end{ of DeleteTreeLeaf };

procedure RecoverTreeLeaf;
  var p1,p2,p,q:advertex;
      v1,v2 : integer;

begin
  v1:=leafval;
  v2:=adjofleaf;
  new(p1); new(p2);
  p1^.int:=v1;
  p2^.int:=v2;
  p1^.next:=tree[v2];
  tree[v2]:=p1;
  degree[v2]:=degree[v2]+1;
  p2^.next:=tree[v1];
  tree[v1]:=p2;
  degree[v1]:=degree[v1]+1;
  if degree[adjofleaf]=2 then
           { adjofleaf is not leaf now! }
    begin
      NumLeaves:=NumLeaves-1;
      p:=head;
      if adjofleaf=p^.int then
         begin
         head:=p^.next;
         dispose(p);
      end
      else
```

```
        begin
        while p^.int<>adjofleaf do
          begin
            q:=p;
            p:=p^.next;
          end;
        q^.next:=p^.next;
        dispose(p);
      end {if};
   end {if}
end{ of RecoverTreeLeaf };

function GetNextOf(fixelem:integer):integer;
  var found: boolean;
          i: integer;

begin
  found:= false;
  i:=fixelem+direction;
  while not(found) do
  begin
    if i=nn+1  then i:=1;
    if i=0 then i:=nn;
    if usedelement[i]=true
    then i:=i+direction
    else found:=true;
  end;
  GetNextOf:=i;
end{ GetNextOf };

begin
  if n=4 then Deal4(despos,srcpos)
  else
  begin
    srcelement:=space[srcpos];
    fixpos:=FindDeleteLeaf(despos);
```

```
    fixelem:=space[fixpos];
    dpos:=tree[fixpos]^.int;
    tr.pos1:=fixpos;
    tr.pos2:=dpos;
    DeleteTreeLeaf(fixpos);
    direction:=RIGHT;
    next:=GetNextOf(fixelem);
    if srcelement<>next
    then direction:=LEFT;

    i:=fixelem;
    for count:=1 to n-1 do
    begin
      next:=GetNextOf(i);
      curno:=next;
      curpos:=positionof[curno];
      usedelement[i]:=true;
      ListPerm(n-1,dpos, curpos);
      usedelement[i]:=false;
      Interchange(tr);
      i:=next;
    end;
    curno:= srcelement;
    curpos:=positionof[curno];
    usedelement[i]:=true;
    ListPerm(n-1,despos, curpos);
    usedelement[i]:= false;
    RecoverLeafList;
    RecoverTreeLeaf;
  end
end{ ListPerm };

(*==========PreProcessing==========*)

procedure FindBasePath;
  var tree, i:integer;
```

```
procedure BuildHpath4(bgraph:1..2;pathid:1..4);
var
  stack : array [1..24] of permutation;
  hptemp: array [1..23] of transposition;
  i     : integer;
  done  : array [0..23] of boolean;
  p     : permutation;
  index : permutation;
  empty : boolean;

function Rank ( var p : permutation ) : integer;
var f,i,j,c,r : integer;
begin
  r := 0;
  f := 1;
  for i := 2 to 4 do begin
     c := 0;
     for j := 1 to i-1 do
        if p[j] > p[i] then c := c + 1;
     r := r + c*f;
     f := f*i;
  end;
  Rank := r;
end {of Rank};

procedure back ( k : integer );
var r,i,j,num,t : integer;
begin

  if ( ((k <= 24) or
     (stack[24,1]<>index[1]) or
     (stack[24,2]<>index[2]) or
     (stack[24,3]<>index[3]) or
     (stack[24,4]<>index[4])
     )
     and empty) then
```

```
  begin
    r := Rank( p );
    if not done[r] then
    begin
      done[r] := true;
      stack[k] := p;
      for num:=1 to 3 do
      begin
        i:=edge[bgraph][num].pos1;
        j:=edge[bgraph][num].pos2;
        t := p[i];  p[i] := p[j];  p[j] := t;
        hptemp[k].pos1:=i; hptemp[k].pos2:=j;
        back( k+1 );
        t := p[i];  p[i] := p[j];  p[j] := t;
     end;
     done[r] := false;
    end {if};
  end
  else
    if empty then
    begin
      for i:=1 to 23 do
      hpath4[bgraph,pathid,i]:=hptemp[i];
      empty:=false;
    end;
end{of back};

procedure InitEdge;
  var tr:transposition;

begin
  tr.pos1:=1;  tr.pos2:=2;
  edge[LINE][1]:=tr;
  tr.pos1:=2;  tr.pos2:=3;
  edge[LINE][2]:=tr;
```

```
  tr.pos1:=3;  tr.pos2:=4;
  edge[LINE][3]:=tr;
  tr.pos1:=1;  tr.pos2:=2;
  edge[STAR][1]:=tr;
  tr.pos1:=1;  tr.pos2:=3;
  edge[STAR][2]:=tr;
  tr.pos1:=1;  tr.pos2:=4;
  edge[STAR][3]:=tr;
end{ InitEdge };

begin
  InitEdge;
  for i := 1 to 4 do
  index[i]:=pathindex[pathid,i];
  for i := 0 to 23 do done[i] := false;
  for i := 1 to 4 do  p[i] := i;
  empty:=true;
  back( 1 );
end {BuildHpath4};

begin
  for i:=1 to 4 do          { p1: 1 2 4 3 }
  begin                     { p2: 2 1 3 4 }
     pathindex[i,1]:=i;     { p3: 3 4 2 1 }
     pathindex[i,3]:=5-i;   { p4: 4 3 1 2 }
  end;
  pathindex[1,2]:=2;
  pathindex[2,2]:=1;
  pathindex[3,2]:=4;
  pathindex[4,2]:=3;
  for i:=1 to 4 do
  pathindex[i,4]:=pathindex[5-i,2];
  writeln('PreProcessing... ');
  for tree:=LINE to STAR do
    for i:= 1 to 4 do
```

```
    BuildHpath4(tree, i);
end{of FindBasePath};

procedure CreateTree;
  var  v1,v2:integer;
       p1,p2:advertex;

begin
  for i:=1 to nn do
  begin degree[i]:=0; tree[i]:=nil end;
  for i:=1 to nn-1 do
  begin
    writeln('input edges of the tree');
    readln(v1,v2);
    new(p1); new(p2);
    p1^.int:=v1;
    p2^.int:=v2;
    p1^.next:=tree[v2];
    tree[v2]:=p1;
    degree[v2]:=degree[v2]+1;
    p2^.next:=tree[v1];
    tree[v1]:=p2;
    degree[v1]:=degree[v1]+1;
  end
end{ CreateTree };

procedure CreateLeafList;
  var  i:integer;
       p:advertex;

begin
  head:=nil;
  NumLeaves:=0;
  for i:=1 to nn do
    if degree[i]=1 then
```

```
      begin
        NumLeaves:=NumLeaves+1;
        new(p);
   p^.int:=i;
   p^.next:=head;
   head:=p
      end;
end{ of CreateLeafList };

begin
  FindBasePath;
  writeln('input nn>=4');
  readln(nn);
  writeln('input desposition' );
  readln(desposition);
  writeln('input srcposition' );
  readln(srcposition);
  count:=0;
  for i:=1 to nn do
  begin
    space[i]:=i;
    positionof[i]:=i;
  end;
  CreateTree;
  CreateLeafList;
  ListPerm(nn,desposition,srcposition);
end.
```

VITA

Surname:	Jiang	Given Names:	Ming
Place of Birth:	JiangSu, China	Date of Birth:	Mar 30, 1966

Educational Institutions Attended:

University of Victoria, Canda	Dec. 1990 to Aug. 1992
Univ. of Sci. and Tech. of China Grad. School and Inst. of Software, Academia Sinica	Sept. 1988 to Dec. 1990
Fu Zhou University, China	Sept. 1984 to June 1988

Degree Awarded:

B.S Fu Zhou University 1988

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis: Hamiltonicity of Certain Vertex Symmetric Graphs

Author: 

Ming Jiang

(Name in Block Letters)

July 30, 1992

(Date)