July 2020

Some Results on Seymour's Second-Neighborhood Conjecture and on Decompositions of Graphs

Farid Bouya

SOME RESULTS ON SEYMOUR'S SECOND-NEIGHBORHOOD CONJECTURE AND ON DECOMPOSITIONS OF GRAPHS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Mathematics

by
Farid Bouya
B.S. in Computer Science, Sharif University of Technology–Tehran, Iran, 2012
M.S. in Mathematics, Louisiana State University, 2014
August 2020

For Mom and Dad, who always believed in me.

Always remember that it is impossible to speak in such a way that you cannot be misunderstood: there will always be some who misunderstand you.

—Karl Popper

Unended Quest: An Intellectual Autobiography

Acknowledgments

First and foremost, praises to God, the Almighty, without Whose blessings I would have been nothing.

I would like to express my deep and sincere gratitude to my advisor Professor Bogdan Oporowski for supporting me through thick and thin. His vision, patience, sincerity, and humor kept me motivated and moving throughout my PhD years. After Bogdan, Professor James Oxley helped me the most as I tried to settle into the LSU environment. Thank you James, for the Communicating Math classes, for the detailed feedbacks on my lectures, for the occasional funny moments, and much more. I also thank Professors Guoli Ding, William Adkins, Karl Mahlburg, Mark Davidson, and Alison Grifa for helping me throughout my LSU years.

I would like to send a special thanks to Dr. Pattie Doody for helping me selflessly with my communication skills. Without her help, I would not have been successful in my talks, presentations, or my defense.

I thank all my K-12 math teachers, specially Hosein Kord, who with his passionate Discrete Math teaching made me very serious about Graph Theory. I also thank my undergraduate advisor Professor Ebadollah Mahmoodian for always being available to help, even after retirement.

I am very much thankful to my wife Saeideh for her patience, understanding, and unconditional support. I also am extremely grateful to my parents and my brother and sister for their financial and emotional support since day one.

Last but not least, I would like to thank all my friends, specially Arash "Doctor"

Ardehali, Amir “Hazrat” Vaziri, Soroush Mojtahedi, Joshua Fallon, and Tara Fife, for their nonstop motivational and spiritual support.

Contents

Abstract

This dissertation consists of two parts. In the first part, I examine Seymour's Second-Neighborhood Conjecture, which states that every orientation of every simple graph has at least one vertex v such that the number of vertices of out-distance 2 from v is at least as large as the number of vertices of out-distance 1 from it. I present alternative statements of this conjecture using the language of linear algebra, the last one being completely in terms of the inverse of some matrix. In the second part of this dissertation, comprising of Chapters 2 and 3, I examine two conjectures on graph decompositions. The first one proposes that every even order hypercube Q_{2n} has a symmetric Hamilton decomposition, meaning that every cycle can be derived from every other cycle just by permuting the axes. I show that this conjecture holds when n is of the form $2^a 3^b$. The second conjecture states that for every graph G its edge set can be partitioned into two sets E_1 and E_2 such that the contractions G/E_1 and G/E_2 are K_4-minor free. This conjecture is currently open, but I ask and answer two slightly different questions: If I use three sets in the partition, contracting two sets at a time, I can avoid K_4 as a minor, but if I use two sets in the partition, contracting one set at a time, there are some graphs that force a $K_{2,3}$ minor.

Chapter 1. Second-Neighborhood Conjecture

1.1. Introduction and Basic Definitions

In this chapter, all directed graphs, or digraphs for short, have underlying graphs that are simple, that is, with no loops and no multiple edges. Let D be a digraph and let u and v be vertices of D. We write $d(u,v)$ to denote the length of the shortest directed path from u to v; if no such path exists, then we put $d(u,v) = \infty$. Since we focus on vertices of out-distance one or two from a particular vertex v of D, we set up the following notation.

$$N^{+}(v) = \{u \in V(D) \mid d(v,u) = 1\}, \qquad d^{+}(v) = |N^{+}(v)|,$$

$$N^{++}(v) = \{u \in V(D) \mid d(v,u) = 2\}, \qquad d^{++}(v) = |N^{++}(v)|,$$

$$N^{-}(v) = \{u \in V(D) \mid d(u,v) = 1\}, \qquad d^{-}(v) = |N^{-}(v)|,$$

$$N^{--}(v) = \{u \in V(D) \mid d(u,v) = 2\}, \qquad d^{--}(v) = |N^{--}(v)|.$$

Each of the symbols defined above may also have a subscript indicating to which digraph it refers. Let $\overleftarrow{D}$ be the digraph obtained from D by reversing the direction on all its edges, so that $d^{+}_{D}(v) = d^{-}_{\overleftarrow{D}}(v)$. The original form of Seymour's Second-Neighborhood Conjecture (SNC) is therefore stated as:

Conjecture 1.1.1 (Seymour; 1990)**.** *Every digraph has a vertex v for which $d^{+}(v) \leqslant d^{++}(v)$.*

One of the most important conjectures regarding digraphs is the famous Caccetta-Häggkvist Conjecture [6]:

This chapter is adapted from: Bouya, F. and Oporowski, B., Seymour's second-neighborhood conjecture from a different perspective, *arXiv pre-print* (2019). It is reprinted with permission from arXiv.

Conjecture 1.1.2 (Caccetta, Häggkvist; 1978)**.** *Every digraph on n vertices that satisfies $d^+(v) \geqslant r$ for all its vertices has a directed cycle with length at most $\lceil \frac{n}{r} \rceil$.*

Conjecture 1.1.1, if true, settles a special case of Conjecture 1.1.2.

We will adopt some of the notation common in linear algebra. In particular, $\mathbf{0}$ will denote a vector or a matrix consisting of all zeros, and similarly, $\mathbf{1}$ will denote a vector or a matrix consisting of all ones. The identity matrix will be denoted by I. Even though the dimensions of these matrices or vectors will not be stated explicitly, they may be easily inferred from the context.

When vectors are represented in the matrix form, they will be understood as column vectors, but to save space, they will be written as transpositions of row vectors. Let $\mathbf{u} = (u_1, u_2, \ldots, u_n)^\top$ and let $\mathbf{v} = (v_1, v_2, \ldots, v_n)^\top$. When we express a numerical relation between vectors, such as $\mathbf{u} \leqslant \mathbf{v}$, we mean that $u_i \leqslant v_i$ for all i in $\{1, 2, \ldots, n\}$. The relations $<$, $\geqslant$, $>$, and $=$ are understood in a similar way. However, the negated relations, such as $\nleqslant$, $\nless$, $\ngeqslant$, $\ngtr$, and $\neq$ are understood in a different way. When we write, for example, $\mathbf{u} \nleqslant \mathbf{v}$ we mean that $u_i > v_i$ for at least one i in $\{1, 2 \ldots, n\}$, and so for vectors with more than one component, the inequality $\mathbf{u} \leqslant \mathbf{v}$ is not equivalent to $\mathbf{u} \ngtr \mathbf{v}$. The same idea applies to all other negated relations.

A *weight function* on a digraph D is a function $w : V(D) \to [0, \infty)$. If the vertices of D are enumerated as v_1, v_2, $\ldots$, v_n, then we can treat w as a vector: $\mathbf{w} = [w(v_1), w(v_2), \ldots, w(v_n)]^\top$. In fact, we will often blur the distinction between the values of a weight function and the components of the vector it determines, and write $\mathbf{w}(v)$ instead of $w(v)$. We will extend this notation to sets of vertices and write $\mathbf{w}(S)$ to mean $\sum_{v \in S} \mathbf{w}(v)$ for a subset S of $V(D)$.

In order to write SNC in terms of matrices, we define the *second-neighborhood matrix of D* as an $n \times n$ matrix S_D whose entries are denoted by s_{ij} and defined as follows:

$$s_{ij} = \begin{cases} 1 & d(v_i, v_j) = 1, \\ -1 & d(v_i, v_j) = 2, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $S_D^{\intercal}$ is the second-neighborhood matrix of $\overleftarrow{D}$.

Example 1.1.3. *Below we have given a digraph D together with its second-neighborhood matrix S_D.*

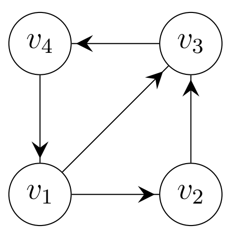


Figure 1.1

$$S_D = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

In this chapter, we have adopted main proof techniques from a paper of Fisher [11].

1.2. Conjectures

The main purpose of this chapter is to present several statements in the language of linear algebra, each of which is equivalent to SNC, in the hope that the tools of linear algebra may yield themselves to attacking the conjecture. These statements are the following:

Conjecture 1.2.1. *Every digraph D satisfies $S_D\mathbf{1} \ngeq \mathbf{0}$.*

Conjecture 1.2.2. *Every digraph D and every weight vector $\mathbf{w}$ on D satisfy $S_D\mathbf{w} \ngeq \mathbf{0}$.*

Conjecture 1.2.3. *For every digraph D there is a non-zero weight vector $\mathbf{w}$ with*

$$S_D\mathbf{w} \leqslant \mathbf{0}.$$

Conjecture 1.2.4. *For every digraph D, there is a vector $\mathbf{v}$ (not necessarily a weight vector) with at least one positive component and such that $S_D\mathbf{v} \leqslant \mathbf{0}$.*

Conjecture 1.2.5. *There is no digraph D such that $S_D^{-1} \geqslant \mathbf{0}$.*

We illustrate how these conjectures might be true in the following example.

Example 1.2.6. *For the digraph D and its second-neighborhood matrix given in Example 1.1.3 we have*

$$S_D\mathbf{1} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \not\leqslant \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

If we pick some weight vector, say $\mathbf{w} = [0, 3, 1, 2.5]^\intercal$, we get

$$S_D\mathbf{w} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 1 \\ 2.5 \end{bmatrix} = \begin{bmatrix} 1.5 \\ -1.5 \\ 2.5 \\ -4 \end{bmatrix} \not\leqslant \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Putting $\mathbf{w} = [1, 0, 1, 1]^\intercal$ gives

$$S_D\mathbf{w} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \\ 1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leqslant \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The same vector also shows that S_D is not invertible, so $S_D^{-1} \not\geqslant \mathbf{0}$ is vacuously true.

The first major result of this chapter is the following:

Theorem 1.2.7. *Conjectures 1.1.1, 1.2.1, 1.2.2, 1.2.3, 1.2.4, and 1.2.5 are equivalent.*

Proving some of the equivalences is significantly harder than proving others, and, indeed, some of these statements, such as Conjectures 1.2.1 and 1.2.2 play only auxiliary roles in the arguments. The proof of this theorem will be presented in a series of propositions in future sections.

If one, and thus all, of these conjectures fail, the sets of counterexamples may, and, in fact, do differ between some of them. When we compare potential counterexamples and use words like "minimal" or "smaller", we understand them in terms of the number of arcs. The fact that the sets of minimal counterexamples to Conjectures 1.2.3, 1.2.4, and 1.2.5 are the same can be easily seen from the proofs of the relevant equivalences. However, we find surprising the following:

Theorem 1.2.8. *Every minimal counterexample to Conjecture 1.2.3 is smaller than every minimal counterexample to Conjecture 1.2.1.*

1.3. Equivalences

We begin by addressing the equivalence of the first pair of the conjectures. We state it without proof, as it is evident.

Proposition 1.3.1. *Conjectures 1.1.1 and 1.2.1 are equivalent.*

We proceed now to the equivalence of the next pair of conjectures.

Proposition 1.3.2. *Conjectures 1.2.1 and 1.2.2 are equivalent.*

Proof. It is clear that Conjecture 1.2.2 implies Conjecture 1.2.1.

Suppose now that Conjecture 1.2.2 fails, and so there are a digraph D and a weight vector $\mathbf{w}$ on D are such that $S_D\mathbf{w} > \mathbf{0}$. Since the set of positive rational numbers forms a dense subset of $[0, \infty)$, we may take a weight vector $\mathbf{w}'$ sufficiently close to $\mathbf{w}$ so that the components of $\mathbf{w}'$ are rational and positive, and $S_D\mathbf{w}' > \mathbf{0}$. By multiplying $\mathbf{w}'$ by a suitable integer, we obtain a weight vector $\mathbf{u}$ whose components are positive integers, and such that $S_D\mathbf{u} > \mathbf{0}$.

We construct a digraph D^* as follows. Enumerate the vertices of D as v_1, v_2, ..., v_n, and suppose that $\mathbf{u} = [u(v_1), u(v_2), \ldots, u(v_n)]^\intercal$. For each i in $\{1, 2, \ldots, n\}$, let V_i be a set of $u(v_i)$ elements, and let $V(D^*)$ be the disjoint union of all V_i's. For each directed edge (v_i, v_j) of D, put into D^* directed edges from each element of V_i to each element of V_j. Let S_{D^*} be the second-neighborhood matrix of D^* and note that in the vector $S_{D^*}\mathbf{1}$, the component corresponding to a vertex v of D^* that lies in in some V_i is equal to the component of $S_D\mathbf{u}$ corresponding to the vertex v_i of D. Hence $S_{D^*}\mathbf{1} > \mathbf{0}$, and so D^* is a counterexample to Conjecture 1.2.1. □

Our proof of the next equivalence will make use of a classical result in linear algebra, known as Farkas' Lemma, which is stated below.

Theorem 1.3.3 (Farkas' Lemma). *Let M be an $(m \times n)$-matrix and let $\mathbf{b}$ be an m-dimensional vector. Then exactly one of the following statements holds.*

1. *There is an n-dimensional vector $\mathbf{x}$ such that $M\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geqslant \mathbf{0}$.*
2. *There is an m-dimensional vector $\mathbf{y}$ such that $M^\intercal\mathbf{y} \geqslant \mathbf{0}$ and $\mathbf{b}^\intercal\mathbf{y} < 0$.*

Proposition 1.3.4. *Conjectures 1.2.3 and 1.2.2 are equivalent. Moreover, a digraph D*

is a counterexample to Conjecture 1.2.3 if and only if $\overleftarrow{D}$ is a counterexample to Conjecture 1.2.2.

Proof. Suppose D is digraph on n vertices. Construct a new matrix M with $n+1$ rows and $2n$ columns by assembling together smaller matrices, as follows:

$$M = \left[\begin{array}{c|c} S_D & I \\ \hline \mathbf{1}^\intercal & \mathbf{0}^\intercal \end{array}\right],$$

and let $\mathbf{b}$ be the $(n+1)$-dimensional standard basis vector $[0, 0, \ldots, 0, 1]^\intercal$.

For the remainder of the proof, we present a list of statements (1)–(9) that are equivalent to one another. It is easy to see that consecutive statements are equivalent, and we remark that the equivalence between (5) and (6) follows from Theorem 1.3.3.

1. Digraph D is a counterexample to Conjecture 1.2.3.
2. The following system fails for every n-dimensional vector $\mathbf{w}$.

$$\begin{cases} S_D\mathbf{w} \leqslant \mathbf{0} \\ \mathbf{w} \geqslant \mathbf{0} \\ \mathbf{w} \neq \mathbf{0} \end{cases}$$

3. The following system fails for every n-dimensional vector $\mathbf{u}$.

$$\begin{cases} S_D\mathbf{u} \leqslant \mathbf{0} \\ \mathbf{u} \geqslant \mathbf{0} \\ \mathbf{1}^\intercal\mathbf{u} = 1 \end{cases}$$

4. The following system fails for every two n-dimensional vectors $\mathbf{u}$ and $\mathbf{z}$.

$$\begin{cases} S_D\mathbf{u} + \mathbf{z} = \mathbf{0} \\ \mathbf{z} \geqslant \mathbf{0} \\ \mathbf{u} \geqslant \mathbf{0} \\ \mathbf{1}^\top\mathbf{u} = 1 \end{cases}$$

5. The following system fails for every $2n$-dimensional vector $\mathbf{x}$.

$$\begin{cases} M\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geqslant \mathbf{0} \end{cases}$$

6. There is an $(n+1)$-dimensional vector $\mathbf{y}$ that satisfies the following system.

$$\begin{cases} M^\top\mathbf{y} \geqslant \mathbf{0} \\ \mathbf{b}^\top\mathbf{y} < 0 \end{cases}$$

7. There are an n-dimensional vector $\mathbf{p}$ and a scalar r that satisfy the following system.

$$\begin{cases} S_D^\top\mathbf{p} + r\mathbf{1} \geqslant \mathbf{0} \\ \mathbf{p} \geqslant \mathbf{0} \\ r < 0 \end{cases}$$

8. There is an n-dimensional vector $\mathbf{p}$ that satisfies the following system.

$$\begin{cases} S_D^\top\mathbf{p} > \mathbf{0} \\ \mathbf{p} \geqslant \mathbf{0} \end{cases}$$

9. Digraph $\overleftarrow{D}$ is a counterexample to Conjecture 1.2.2.

We established the equivalence of statements (1) and (9), which concludes the proof. □

Next we show that Conjectures 1.2.4 and 1.2.5 are equivalent.

Proposition 1.3.5. *Conjectures 1.2.4 and 1.2.5 are equivalent, with the same set of counterexamples.*

Proof. Let D be a digraph, and suppose first that the matrix S_D is not invertible. Then there is a non-zero vector $\mathbf{u}$ such that $S_D\mathbf{u} = \mathbf{0}$. If $\mathbf{u}$ has a positive component, then let $\mathbf{v} = \mathbf{u}$; otherwise let $\mathbf{v} = -\mathbf{u}$. Then $\mathbf{v}$ testifies to the fact that D satisfies Conjecture 1.2.4. Also, D vacuously satisfies Conjecture 1.2.5, and so both conjectures hold for digraphs with non-invertible second-neighborhood matrices.

Suppose now that S_D is invertible, and let σ_D be the map defined by $\sigma_D\colon \mathbf{w} \mapsto S_D\mathbf{w}$. Consider the statement:

(1) Digraph D is a counterexample to Conjecture (1.2.4).

It is equivalent to the statement that no vector $\mathbf{w}$ satisfies both $S_D\mathbf{w} \leqslant \mathbf{0}$ and $\mathbf{w} \nleqslant \mathbf{0}$, which, in turn, is equivalent to the statement:

(2) If $\sigma_D(\mathbf{w}) \leqslant \mathbf{0}$, then $\mathbf{w} \leqslant \mathbf{0}$.

Since S_D is invertible, σ_D is bijective and thus has an inverse, and so statement (2) is equivalent to the following:

(3) If $\mathbf{w} \leqslant \mathbf{0}$, then $\sigma_D^{-1}(\mathbf{w}) \leqslant \mathbf{0}$.

Note that σ_D is also linear, and so (3) is equivalent to the statement:

(4) If $\mathbf{w} \geqslant \mathbf{0}$, then $\sigma_D^{-1}(\mathbf{w}) \geqslant \mathbf{0}$.

Now, we observe that $\sigma_D^{-1}(\mathbf{w}) = S_D^{-1}\mathbf{w}$, and so (4) may be restated as:

(5) $S_D^{-1}\mathbf{w} \geqslant \mathbf{0}$ for every vector $\mathbf{w} \geqslant \mathbf{0}$.

The last statement holds if and only if every entry of S_D^{-1} is non-negative. □

The last equivalence is established in the next section.

1.4. Counterexamples

In this section, we will compare the various sets of potential counterexamples to the conjectures discussed in this chapter.

For each N in $\{1.2.1, 1.2.3, 1.2.4\}$, let $\mathcal{X}_N$ denote the set of counterexamples to Conjecture N, and let $\overleftarrow{\mathcal{X}}_N = \{\overleftarrow{D} \mid D \in \mathcal{X}_N\}$. Intuitively, we may think of each $\overleftarrow{\mathcal{X}}_N$ as the set of counterexamples to "Conjecture N stated for in-neighbors".

The first proposition comparing the above sets of counterexamples is an immediate consequence of the statements of the conjectures, so it is stated without proof.

Proposition 1.4.1. $\mathcal{X}_{1.2.4} \subseteq \mathcal{X}_{1.2.3}$ *and* $\overleftarrow{\mathcal{X}}_{1.2.4} \subseteq \overleftarrow{\mathcal{X}}_{1.2.3}$.

The next proposition is almost as obvious.

Proposition 1.4.2. $\mathcal{X}_{1.2.1} \subseteq \overleftarrow{\mathcal{X}}_{1.2.3}$.

Proof. Suppose $D \in \mathcal{X}_{1.2.1}$. It is obvious that D is also a counterexample to Conjecture 1.2.2, and Proposition 1.3.4 asserts that $\overleftarrow{D}$ is a counterexample to Conjecture 1.2.3, as well; the conclusion follows. □

Lemma 1.4.3. *Every minimal element of* $\mathcal{X}_{1.2.3}$ *is a member of* $\mathcal{X}_{1.2.5}$.

Proof. Let D be a minimal element of $\mathcal{X}_{1.2.3}$, and let $V(D) = \{v_1, v_2, \ldots, v_n\}$. By the minimality of D, for each $D - v_i$ there is a non-zero non-negative weight vector $\mathbf{w}_i$ satisfying $S_{D-v_i}\mathbf{w}_i \leqslant \mathbf{0}$. We can extend $\mathbf{w}_i$ to a weight vector $\mathbf{w}_i$ on D by putting $\mathbf{w}_i(v_i) = 0$. Note that $\mathbf{w}_i(N^+(v_j)) - \mathbf{w}_i(N^{++}(v_j)) \leqslant 0$ for $j \neq i$. If for some i, the weight vector $\mathbf{w}_i$ satisfies $S_D\mathbf{w}_i \leqslant \mathbf{0}$, then we reach a contradiction. Therefore, we may assume that $\mathbf{w}_i(N^+(v_i)) - \mathbf{w}_i(N^{++}(v_i)) > 0$ for all i. Let $\widehat{W}$ be the $(n \times n)$-matrix whose ith column is $\mathbf{w}_i$, and let $C = S_D\widehat{W}$. Then the entries of C may be expressed as $c_{ij} = \mathbf{w}_j(N^+(v_i)) - \mathbf{w}_j(N^{++}(v_i))$, which implies that c_{ij} is positive if and only if $i = j$.

We use a process similar to the Gauss-Jordan elimination to turn C into the identity matrix I_n. The only difference is that we work with columns instead of rows, so we do elementary column operations. If we are successful, the identity matrix I_n may be expressed as C multiplied on the right by an appropriate transformation matrix T, that is, $I_n = CT$. To be more precise, we do the following:

1. Start by putting $i = 1$ and $X = (x_{ij}) = C$.
2. If $i > n$, then X is equal to I_n. Exit.
3. If $x_{ii} \leqslant 0$, exit. Otherwise, add suitable multiples of the ith column of X to other columns of X to make the ith row of X zero (except for x_{ii}).
4. Divide the ith column by x_{ii}.
5. Add 1 to i. Go to (2).

If during this process we get non-positive ith diagonal (that is, the algorithm exits through step (3) because $x_{ii} \leqslant 0$), then a non-negative, non-zero linear combination of

$S_D\widehat{\mathbf{w}}_1$, $S_D\widehat{\mathbf{w}}_2$, $\ldots$, $S_D\widehat{\mathbf{w}}_i$ is non-positive, say,

$$a_1S_D\widehat{\mathbf{w}}_1 + a_2S_D\widehat{\mathbf{w}}_2 + \cdots + a_iS_D\widehat{\mathbf{w}}_i \leqslant \mathbf{0}.$$

This is equivalent to $S_D\left(a_1\widehat{\mathbf{w}}_1 + a_2\widehat{\mathbf{w}}_2 + \cdots + a_i\widehat{\mathbf{w}}_i\right) \leqslant \mathbf{0}$, which contradicts the fact that $D \in \mathcal{X}_{1.2.3}$. Therefore the procedure described above never results in the matrix X having a non-positive entry on the main diagonal, so the algorithm never exits through step (3), and always exits through step (2) instead, giving us the identity matrix I_n. Note that in this process, we only add non-negative multiples of a column to other columns. This means that the elementary matrices associated with the matrix operations are all non-negative, therefore their product T is also non-negative. Let $W' = \widehat{W}T$, let $\mathbf{w}'_i$ be the ith column of W', and let $\mathbf{e}_i$ be the ith column of I_n, that is, the ith n-dimensional standard basis vector. Then W' is non-negative. We have

$$I_n = CT = S_D\widehat{W}T = S_DW'.$$

This means that S_D has non-negative inverse, so $D \in \mathcal{X}_{1.2.5}$, as required. $\square$

Now we are ready to provide the last part of the proof of Theorem 1.2.7.

Proposition 1.4.4. *Conjectures 1.2.3 and 1.2.4 are equivalent.*

Proof. Clearly, Conjecture 1.2.3 implies Conjecture 1.2.4.

Suppose now that Conjecture 1.2.3 fails, and so some digraph D is a minimal element of $\mathcal{X}_{1.2.3}$. Lemma 1.4.3 implies that $D \in \mathcal{X}_{1.2.5}$, so Conjecture 1.2.5 fails. Proposition 1.3.5 now implies that Conjecture 1.2.4 fails as well. $\square$

The remainder of the chapter is devoted to proving Theorem 1.2.8. Most of the work will be contained in the following:

Lemma 1.4.5. *If a digraph D is a minimal member of $\mathcal{X}_{1.2.3}$, then $D \notin \overleftarrow{\mathcal{X}}_{1.2.1}$.*

Proof. Suppose, for a contradiction, that D is a minimal member of $\mathcal{X}_{1.2.3}$ that also belongs to $\overleftarrow{\mathcal{X}}_{1.2.1}$. Since Proposition 1.4.2 asserts that $\overleftarrow{\mathcal{X}}_{1.2.1} \subseteq \mathcal{X}_{1.2.3}$, we also have

(1) D is a minimal element of $\overleftarrow{\mathcal{X}}_{1.2.1}$.

The minimality of D in $\overleftarrow{\mathcal{X}}_{1.2.1}$ implies that it is strongly connected, and the fact that $\overleftarrow{D}$ is a counterexample to SNC implies that the minimum in-degree of D is at least two; in fact it is at least seven (see [13]).

Let y be an arbitrary vertex of D, let xy be an arc of D, and let $D' = D \setminus xy$. For a vertex v of D, let $a(v) = d^-_D(v) - d^{--}_D(v)$ and let $a'(v) = d^-_{D'}(v) - d^{--}_{D'}(v)$. Note that $a(v) \leqslant a'(v)$ whenever $v \neq y$. If D has a directed path of length two from x to y, then $a'(y) = a(y) - 2$; otherwise $a'(y) = a(y) - 1$. We show that

(2) $a'(y) = -1$ and $a'(v) \geqslant 1$ for $v \neq y$.

It is not hard to see that $a(v) \in \{1, 2\}$; see [5] for a justification. This means that $a'(y) \in \{-1, 0, 1\}$ and $a'(v) \geqslant 1$ for $v \neq y$. In the case $a'(y) = 1$, we reach a contradiction with the minimality of D in $\overleftarrow{\mathcal{X}}_{1.2.1}$. We will show that $a'(y) = 0$ cannot occur either.

Suppose, for a contradiction, that $a'(y) = 0$, and let z be a vertex in $N^-_{D'}(y)$. We define a weight vector $\mathbf{u}$ on D' as follows:

$$\mathbf{u}(v) = \begin{cases} 1 & \text{if } v \neq z; \text{ and} \\ \frac{3}{2} & \text{if } v = z; \end{cases}$$

Now, we have $S^{\top}_{D'}\mathbf{u} > \mathbf{0}$, and an argument very similar to the proof of Proposition 1.3.4 im-

plies that D' fails Conjecture 1.2.3, which contradicts the minimality of D in $\mathcal{X}_{1.2.3}$. Thus we conclude that $a'(y) = -1$.

Since $D \in \overleftarrow{\mathcal{X}}_{1.2.1}$, it satisfies $a(y) > 0$, and thus, it must be that $a(y) = 1$. In other words,

(3) $d^-_D(y) = d^{--}_D(y) + 1$.

Let y be a vertex of D with the largest possible in-degree d. Then (3) implies that $d^{--}(y) = d - 1$. Let $N^-(y) = X = \{x_1, x_2, \ldots, x_d\}$ and let $N^{--}(y) = Z = \{z_1, z_2, \ldots, z_{d-1}\}$. Consider the digraph $D' = D \setminus x_1 y$ and note that the minimality of D implies that there is a weight vector $\mathbf{w}'$ such that $S_{D'}\mathbf{w}' \leqslant \mathbf{0}$. The last inequality is equivalent to stating that $\mathbf{w}'(N^+(u)) \leqslant \mathbf{w}'(N^{++}(u))$ for every vertex u of D', which, in turn, implies that

$$\sum_{u \in V(D')} \mathbf{w}'(N^+_{D'}(u)) \leqslant \sum_{u \in V(D')} \mathbf{w}'(N^{++}_{D'}(u))$$

Note that $\mathbf{w}'(u)$ appears $d^-_{D'}(u)$ times on the left side of the above inequality, while it appears $d^{--}_{D'}(u)$ times on the right side. By (3), we have $d^-_{D'}(u) \geqslant d^{--}_{D'}(u) + 1$ whenever $u \neq y$ and $d^-_{D'}(y) = d^{--}_{D'}(y) - 1$, and so $\mathbf{w}'(y) \geqslant \mathbf{w}'(V \setminus \{y\})$. If $\mathbf{w}'(y) > \mathbf{w}'(V \setminus \{y\})$, then $\mathbf{w}'(N^+_{D'}(x_2)) > \mathbf{w}'(N^{++}_{D'}(x_2))$, which is impossible. It follows that $\mathbf{w}'(y) = \mathbf{w}'(V \setminus \{y\})$, which implies that $\mathbf{w}'(N^+_{D'}(u)) = \mathbf{w}'(N^{++}_{D'}(u))$ for every vertex u of D'.

Let $S = \{u \in V(D') : u \neq y \text{ and } \mathbf{w}'(u) > 0\}$, and observe that $\mathbf{w}'(S) = \mathbf{w}'(y)$. Let $k = \mathbf{w}'(y)$, let $Z' = Z \cup \{x_1\}$, and let $X' = X \setminus \{x_1\}$. By construction, $N^{--}_{D'}(y) = Z'$, and so $y \in N^{++}_{D'}(z)$ for every $z \in Z'$. This implies that $\mathbf{w}'(N^{++}_{D'}(z)) \geqslant k$, and, further, that $\mathbf{w}'(N^{++}_{D'}(z)) = k = \mathbf{w}'(N^+(z))$. This means that D has an arc zs for every $z \in Z'$ and every $s \in S$. Since y has the largest in-degree in D, we have $Z' = N^-(s)$ for every $s \in S$.

Note that $y \notin S$, and if $X' \cap S$ had an element x, then we would have $\mathbf{w}'(N^{++}(x)) <$

$k \leqslant \mathbf{w}'(N^+(x))$, which is impossible; hence $X' \cap S = \varnothing$. Similarly, $Z' \cap S = \varnothing$. Therefore $(\{y\} \cup N^-(y) \cup N^{--}(y)) \cap S = \varnothing$. Since D is strongly connected and S is non-empty, D has a vertex t of in-distance three from y, which, clearly, is in neither X nor Z. Since all vertices in S have d in-neighbors in $X \cup Z$, there is no arc in D in the form ts with $s \in S$, and so $\mathbf{w}'(N^+(t)) = 0$. But every $z \in N^{--}(y)$ has arcs to all members of S, so $\mathbf{w}'(N^{++}(t)) \geqslant k$; a contradiction. □

Finally, we are ready to prove Theorem 1.2.8

Proof of Theorem 1.2.8. Suppose D is a minimal counterexample to Conjecture 1.2.1. Then $\overleftarrow{D} \in \overleftarrow{\mathcal{X}}_{1.2.1}$. Lemma 1.4.2 implies that $\overleftarrow{\mathcal{X}}_{1.2.1} \subseteq \mathcal{X}_{1.2.3}$, and so $\overleftarrow{D}$ is also a counterexample to Conjecture 1.2.3. If $\overleftarrow{D}$ were minimal, then Lemma 1.4.5 would imply that $D \notin \mathcal{X}_{1.2.1}$, which would be a contradiction. □

Chapter 2. Hamilton Decompositions of Hypercubes

2.1. Introduction

In this section we give a brief and informal history of the problem. Formal definitions are given in the next section.

Hypercubes are widely used in computer architectures in areas like parallel computing [19], multiprocessor systems [8], processor allocation [20], and fault-tolerant computing [1]. Hamilton decomposition (H.D.) of hypercubes is of central importance in the aforementioned areas.

In 1954, Ringel showed that the hypercube Q_n is Hamilton decomposable whenever n is a power of two and posed the problem of whether a similar decomposition exists for all even n [21]. In 1982, Aubert and Schneider showed that every Q_{2n} admits a Hamilton decomposition [2]. Many different algorithms and methods have been used to find explicit Hamilton decompositions for Q_{2n}. Our work is inspired by two such methods. Okuda and Song [18] gave a direct approach for finding Hamilton decompositions for Q_{2n} with $n \leqslant 4$. Mollard and Ramras [15] gave a fast and efficient method of generating and storing Hamilton decompositions when n is a power of two by constructing one special cycle and permuting the axes to obtain the other cycles. We use Okuda and Song's method to continue the work of Mollard and Ramras and extend it to all n of the form 2^a3^b, which is the main result of this chapter, stated formally as Corollary 2.5.7. In Section 2.7, we present Algo-

This chapter is adapted from: Bouya, F., Mahmoodian, E. S., Shokrian, M., and Tefagh, M., A Highly Symmetric Hamilton Decomposition for Hypercubes, *arXiv pre-print* (2020). It is reprinted with permission from arXiv.

rithms 4 and 5 that efficiently construct such decompositions. We conjecture that a similar decomposition exists for every n.

2.2. Notation

The *hypercube of dimension* n, denoted by Q_n, is the graph whose vertices are the 2^n binary strings of length n and two vertices are adjacent if and only if their corresponding strings differ in exactly one bit. The *Cartesian product* of two graphs G and H, denoted by $G \square H$, has vertex set

$$V(G \square H) = \{(u,v) | u \in V(G) \text{ and } v \in V(H)\},$$

and two of its vertices (u,v) and (u',v') are adjacent if and only if

- $u = u'$ and $vv' \in E(H)$, or
- $v = v'$ and $uu' \in E(G)$.

Using this definition, it is not hard to see that

$$\begin{aligned} Q_2 &= C_4, \\ Q_{m+n} &= Q_m \square Q_n, \\ Q_n &= \underbrace{K_2 \square K_2 \square \cdots \square K_2}_{n \text{ times}}, \end{aligned}$$

and

$$Q_{2n} = \underbrace{C_4 \square C_4 \square \cdots \square C_4}_{n \text{ times}}. \tag{2.2.1}$$

As in [22], we use Equation (2.2.1) to make another coordinate system for the vertices of Q_{2n}:

Each vertex is assigned a quaternary string $q_1 q_2 \dots q_n$ of length n, where $q_i \in \{0, 1, 2, 3\}$. There is an edge between two vertices if and only if their labels differ in exactly one position, and in that position, their difference is either 1 or -1 modulo 4. We wish to consider directed cycles, so we assign directions to edges of Q_{2n} as follows: A *dimension-k edge* in Q_{2n} is an edge that connects two vertices whose quaternary labels differ in the kth digit. If a dimension-k edge is directed in the *positive direction*, that is, it is directed from $(x_1, x_2, \dots, x_{k-1}, x_k, x_{k+1}, \dots, x_n)$ towards $(x_1, x_2, \dots, x_{k-1}, x_k + 1 \pmod 4), x_{k+1}, \dots, x_n)$, we denote it by k, and if it is directed in the opposite direction, we denote it by $\overline{k}$. A *Hamilton cycle* in a graph is a cycle visiting all the vertices. A *Hamilton decomposition* of Q_{2n} is a partitioning of its edge set into n disjoint Hamilton cycles. We use the notation given in [18] to show directed cycles: We start from the initial vertex, and simply move in the positive direction of C, writing down the dimension and the direction of the edges we pass. For example, the cycle given in Figure 2.1, with the origin (top left vertex) as its initial vertex, is shown by $2\overline{11}22\overline{1112}11\overline{22}111$.

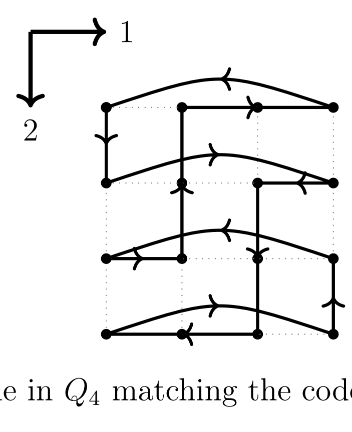


Figure 2.1: The cycle in Q_4 matching the code $2\overline{11}22\overline{1112}11\overline{22}111$.

In this chapter, we only deal with Hamilton cycles, so the initial vertex is always taken to be the origin, that is, the vertex $\mathbf{0} = (0, 0, \ldots, 0)$.

Definition 2.2.1. *Define $G_{n,k}$ to be the graph* $\underbrace{C_{4^n} \square C_{4^n} \square \cdots \square C_{4^n}}_{k \text{ times}}$.

Note that $Q_{2k} \cong G_{1,k}$. We show directed edges and cycles in $G_{n,k}$ the same way we show them in Q_{2n}. The only difference is that coordinates in $G_{n,k}$ are calculated modulo 4^n, whereas they are calculated modulo 4 in Q_{2n}. We are especially interested in the cases $k = 2$ and $k = 3$, so we recognize that these cases require special treatment. Since $G_{n,2}$ is the Cartesian product of two cycles, we think of $G_{n,2}$ as a 2-dimensional cyclic grid. Every vertex of $C_{4^n} \square C_{4^n}$ has coordinates (u, v), where u is in the first copy of C_{4^n} and v is in the second copy. We think of this coordinate (u, v) in two ways:

1. The vertices u and v are elements of Q_{2n}, and thus quaternary strings of length n. Therefore, (u, v) is a quaternary string of length $2n$.
2. Fixing some order in Q_{2n}, we assign the integers 0 to $4^n - 1$ to its vertices. Thus, every vertex in Q_{4n} has integral coordinates (u, v) where $0 \leqslant u, v \leqslant 4^n - 1$.

In order to derive Hamilton decompositions for larger hypercubes from smaller hypercubes, we study the graphs $G_{n,2}$ and $G_{n,3}$ in more detail.

2.3. The 2-Dimensional Case

We start by finding an H.D. for $G_{n,2}$. We will see how an H.D. for $G_{n,2}$ and an H.D. for Q_{2n} can be combined to give an H.D. for Q_{4n}.

2.3.1. An H.D. for $G_{n,2}$

There is an H.D. for the graph $C_m \square C_m$ in

$$H_1 = \underbrace{\underbrace{11\cdots1}_{m-1\text{ times}}\ 2\ \underbrace{11\cdots1}_{m-1\text{ times}}\ 2\cdots\ \underbrace{11\cdots1}_{m-1\text{ times}}\ 2}_{m\text{ times}}\quad,\quad H_2 = \underbrace{\underbrace{22\cdots2}_{m-1\text{ times}}\ 1\ \underbrace{22\cdots2}_{m-1\text{ times}}\ 1\cdots\ \underbrace{22\cdots2}_{m-1\text{ times}}\ 1}_{m\text{ times}}$$

Since $G_{n,2} = C_{4^n} \square C_{4^n}$, we have the same type of H.D. for $G_{n,2}$:

$$H_1 = \underbrace{\underbrace{11\cdots1}_{4^n-1\text{ times}}\ 2\ \underbrace{11\cdots1}_{4^n-1\text{ times}}\ 2\cdots\ \underbrace{11\cdots1}_{4^n-1\text{ times}}\ 2}_{4^n\text{ times}}\quad,\quad H_2 = \underbrace{\underbrace{22\cdots2}_{4^n-1\text{ times}}\ 1\ \underbrace{22\cdots2}_{4^n-1\text{ times}}\ 1\cdots\ \underbrace{22\cdots2}_{4^n-1\text{ times}}\ 1}_{4^n\text{ times}} \tag{2.3.1}$$

2.3.2. Deriving an H.D. for Q_{4n} From an H.D. for Q_{2n}

Noting that Q_{2n} has order 4^n and $Q_{4n} = Q_{2n} \square Q_{2n}$, we propose the following:

Definition 2.3.1. *Let E be a directed Hamilton cycle in Q_{2n}. A* 2-dimensional seating *of Q_{4n} onto $G_{n,2}$ via E, is a representation of the vertices of Q_{4n} by assigning them integral coordinates as follows:*

1. *Consider E and its positive direction. Take $\mathbf{0}$ as the initial vertex. Assign 0 to $\mathbf{0}$, assign 1 to the next vertex in E, and continue until $4^n - 1$ is assigned to the last vertex of E.*
2. *Induce the order of E onto Q_{2n}, so that each vertex has the same order in either graph.*
3. *Using the coordinates in (2), assign coordinates to every member of $Q_{4n} = Q_{2n} \square Q_{2n}$. Put the vertices on the 2-dimensional grid using their coordinates.*

Using the natural order of E, we have mapped the vertices of Q_{4n} onto $G_{n,2}$ and recognized Q_{4n} as a supergraph of $G_{n,2}$. Any subgraph of $G_{n,2}$, therefore, is also a subgraph of Q_{4n}. In particular, if H is a directed Hamilton cycle in $G_{n,2}$, the 2-dimensional directed Hamil-

ton cycle derived from E and H, *denoted by* $f(E,H)$*, is a Hamilton cycle in* Q_{4n} *and is defined in the natural way:*

1. *2-dimensionally seat* Q_{4n} *onto* $G_{n,2}$ *via* E.
2. Q_{4n} *has* $2n$ *axes* $1, 2, \ldots, 2n$*, while* $G_{n,2}$ *has an* x*-axis and a* y*-axis. The axes* $1, 2, \ldots, n$ *are in direction* x *and the axes* $n+1, n+2, \ldots, 2n$ *are in direction* y.
3. $f(E,H)$ *has the same edges in the supergraph* $Q_{2n} \square Q_{2n}$ *as* H *has in the subgraph* $G_{n,2}$.

The following lemma is very useful.

Lemma 2.3.2. *Let* H_1 *and* H_2 *be two disjoint Hamilton cycles in* $G_{n,2}$ *(which form an H.D.) and* E_1 *and* E_2 *be two disjoint Hamilton cycles in* Q_{2n}*. Then the four Hamilton cycles* $F_1 = f(E_1,H_1)$*,* $F_2 = f(E_1,H_2)$*,* $F_3 = f(E_2,H_1)$*, and* $F_4 = f(E_2,H_2)$ *in* Q_{4n} *are pairwise disjoint.*

Proof. It suffices to show that $F_1 = f(E_1,H_1)$ is disjoint from the other three cycles F_2, F_3, and F_4. To achieve this, we 2-dimensionally seat Q_{4n} onto $G_{n,2}$ via E_1. This enables us to see that F_1 and F_2 have all their edges on the grid, whereas F_3 and F_4 have all their edges off the grid. This means that F_1 is disjoint from F_3 and from F_4. Furthermore, F_1 and F_2 represent H_1 and H_2, respectively, and H_1 and H_2 are disjoint, so F_1 and F_2 must be disjoint as well. □

This provides us with a recursive tool to construct Hamilton decompositions.

Corollary 2.3.3. *If* $\{H_1, H_2\}$ *is an H.D. for* $G_{n,2}$ *and* $\{E_1, E_2, \ldots, E_n\}$ *is an H.D. for* Q_{2n}*, then the family* $\{f(E_i, H_j) \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 2\}$ *is an H.D. for* Q_{4n}*. The new*

Hamilton cycles are named F_1, F_2, $\ldots$, F_{2n} *via* $F_j = f(E_j, H_1)$ *and* $F_{j+n} = f(E_j, H_2)$ *for* $1 \leqslant j \leqslant n$.

2.3.3. 2-Dimensional Algorithm

Using the definition of $f(E,H)$, it is not difficult to devise an algorithm for computing an H.D. for Q_{4n}. Algorithm 1, given in Section 2.7, takes an H.D. for $G_{n,2}$ and an $H.D.$ for Q_{2n} as inputs, and outputs an H.D. for Q_{4n}.

2.4. The 3-Dimensional Case

Just like in the 2-dimensional case, finding an H.D. for the graph $G_{n,3}$ is essential for transitioning from Q_{2n} to Q_{6n}. An H.D. for Q_{2n} can be combined with an H.D. for $G_{n,3}$ to give an H.D. for Q_{6n}.

2.4.1. An H.D. for $G_{n,3}$

Compared to the 2-dimensional case, finding an H.D. for $G_{n,3}$ is not easy. Motivated by [18] and [22], we decompose the graph into three 2-factors, and then try to merge the components until we have three Hamilton cycles.

Lemma 2.4.1. *The graph* $G_{n,3}$ *with the partitioning given below decomposes into* 3×4^n *copies of the directed cycle with* 4^{2n} *edges:*

If e *is in direction 1 and is between* (x,y,z) *and* $(x+1,y,z)$*, we direct* e *from*

(x, y, z) *to* $(x+1, y, z)$ *and*

$$e \in Z \qquad \textit{if } x + y + z = -1 \pmod{4^n},$$

$$e \in X \qquad \textit{otherwise.}$$

If e *is in direction 2 and is between* (x, y, z) *and* $(x, y+1, z)$*, we direct* e *from* (x, y, z) *to* $(x, y+1, z)$ *and*

$$e \in X \qquad \textit{if } x + y + z = -1 \pmod{4^n},$$

$$e \in Y \qquad \textit{otherwise.}$$

If e *is in direction 3 and is between* (x, y, z) *and* $(x, y, z+1)$*, we direct* e *from* (x, y, z) *to* $(x, y, z+1)$ *and*

$$e \in Y \qquad \textit{if } x + y + z = -1 \pmod{4^n},$$

$$e \in Z \qquad \textit{otherwise.}$$

We have demonstrated the case $n = 1$ in Section 2.6.

Proof. Choosing an arbitrary vertex v and moving along the edges of X, we can see that v belongs to a unique cycle of length 4^{2n} that is in X. Similarly, it belongs to a unique cycle of length 4^{2n} in Y and another one in Z. There are 4^{3n} vertices in total, so there are 4^n cycles in each of X, Y, and Z, for a total of 3×4^n cycles. □

We wish to merge these cycles together and end up with just three, so that we have an H.D. for $G_{n,3}$. To this end, we introduce two cubes and a merge operation. These cubes and the merge operation were first introduced in [22] and later in [18] to build an H.D. for Q_6. We use them to construct an H.D. for every $G_{n,3}$

Definition 2.4.2. *Let c_X, c_Y, and c_Z, denote the number of (current) connected components of X, Y, and Z, respectively.*

The type-I cube *and the* type-II cube *are given in Figure 2.2. The top left vertex is the* origin of the cube*, that is, the vertex (x, y, z) such that any other vertex (x', y', z') of the cube satisfies $0 \leqslant x - x', y - y', z - z' \leqslant 1 \pmod{4^n}$.*

(a) Type I

(b) Type II

Figure 2.2: The special cube type-I (2.2a) and the special cube type-II (2.2b).

By merging *a type-I cube we replace it with a type-II cube. Note that the vertices maintain their X-, Y-, and Z-degrees during the merge operation.*

The aim of the merge operation is to reduce each of c_X, c_Y, and c_Z by 1. Before starting to merge, we need to make sure that we have enough type-I cubes and that this three-way switch in colors indeed merges six cycles into three. We make a couple of observations.

Observation 2.4.3. *Consider $G_{n,3}$ and decompose it with the method described in Lemma 2.4.1. Then every vertex (x, y, z) with $x + y + z = -1 \pmod{4^n}$ is the origin of a type-I cube.*

Observation 2.4.4. *Figure 2.3 shows that, a single merge operation, done on the decomposition obtained from Lemma 2.4.1, indeed merges six cycles, two in each of X, Y, and Z, into three cycles, one in each of X, Y, and Z.*

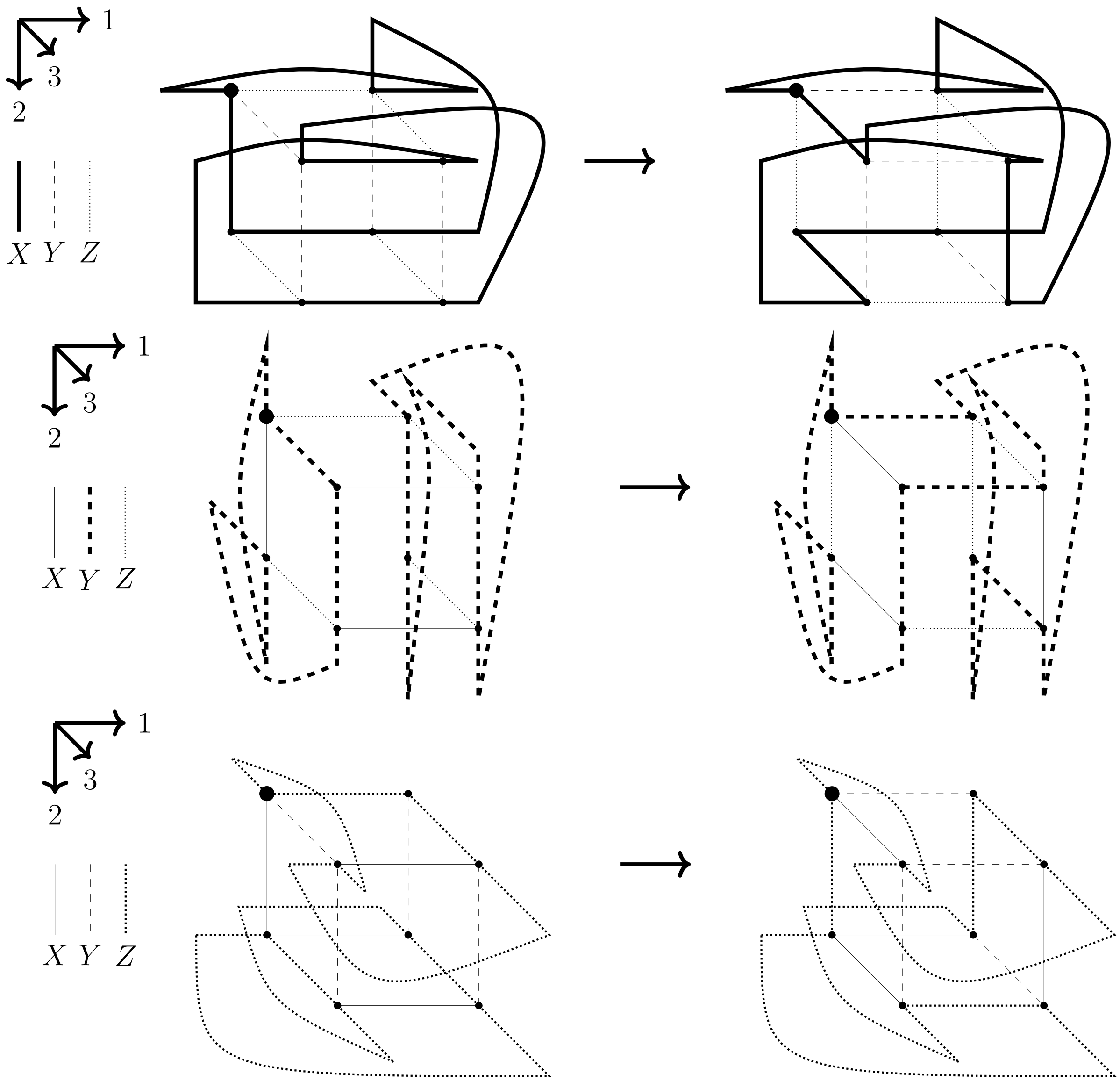


Figure 2.3: The cycles merged during the merge operation

Of course, we need another $4^n - 2$ of these merge operations, and as we progress, the structures of the cycles change, which could possibly cause a merge operation to "fail" to combine six cycles into three. Hence, Lemma 2.4.8 is crucial.

Definition 2.4.5. *For $i \leqslant j$ let $[i,j]$ be the set $\{i, i+1, \ldots, j\}$. For $0 \leqslant i < 4^n$, define Z_i^n to be the set of vertices of $G_{n,3}$ that have their 3rd coordinate equal to i. Finally, let*

$$Z_{[i,j]}^n = \bigcup_{i \leqslant k \leqslant j} Z_k^n$$

Definition 2.4.6. *Let C be a cycle and S be a subset of $V(C)$. The C*-necklace-order with respect to S *is the order in which the vertices of S appear in C. As its name suggests, shifting or reversing the direction of C does not change its order (with respect to any vertex set).*

Observation 2.4.7. *Let $v = (x, y, z)$ be the origin of a type-I cube L. Figure 2.4 shows that the $X \cap Z_x^n$-necklace-order with respect to Z_x^n (before merge) is the same as $X \cap Z_{[x,x+1]}^n$-necklace-order with respect to Z_x^n (after merge). Indeed, the only change to $X \cap Z_x^n$ is the removal of the edge uv, which is replaced by a detour through Z_{x+1}^n. This augmentation does not change the order of vertices of Z_x^n.*

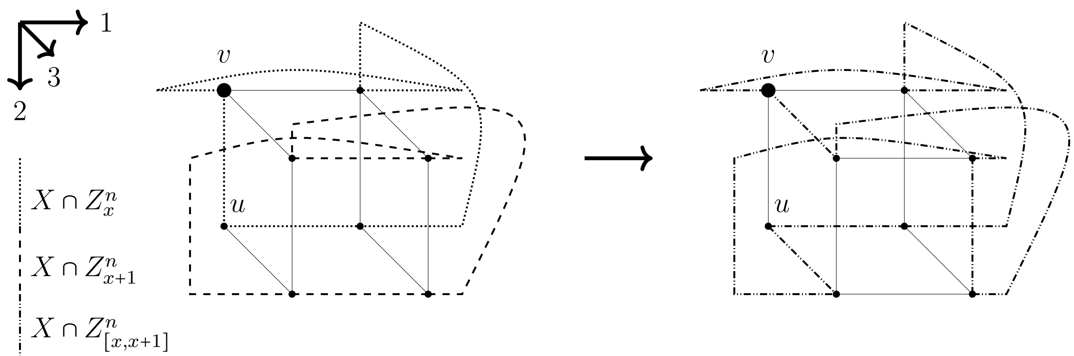


Figure 2.4: The $X \cap Z_x^n$-necklace-order with respect to Z_x^n (left) is the same as the $X \cap Z_{[x,x+1]}^n$-necklace-order with respect to Z_x^n (right).

Lemma 2.4.8. *Suppose that the edge set of $G_{n,3}$ is decomposed with the method given in Lemma 2.4.1, but no switches are performed. Let $v = (x, y, z)$ and $v' = (x', y', z')$ be such that $x + y + z = x' + y' + z' = -1 \pmod{4^n}$ and $z = z' + 1 \pmod{4^n}$, and let L and L' be*

the type-I cubes with origins at v and v', respectively. If we merge L first and then L', we reduce c_X by 2.

Proof. We saw in Observation 2.4.4 that a single merge operation always succeeds. Suppose that we have merged L, so that Z^n_{x+1} and Z^n_{x+2} have merged into $Z^n_{[x+1,x+2]}$, and we are about to merge L'. By Observation 2.4.7, the order of vertices in Z^n_{x+1} has not changed, so merging L' will successfully combine $Z^n_{[x+1,x+2]}$ and Z^n_x into a single cycle $Z^n_{[x,x+2]}$. □

We now specify a condition under which all the merge operations are guaranteed to succeed.

Definition 2.4.9. *Let $S \subseteq [0, 4^n - 1]^3$. We say that S is a* merging set *if it satisfies the following:*

- $|S| = 4^n - 1$,
- *Members (x, y, z) of S satisfy $x + y + z = -1 \pmod{4^n}$, and*
- *Distinct members (x, y, z) and (x', y', z') of S satisfy $x \neq x'$, $y \neq y'$, and $z \neq z'$.*

We need $4^n - 1$ merge operations, each merging six cycles into three. In order for all these operations to successfully take place, it suffices for the type-I cubes to have their origins in a merging set. This we show next.

Lemma 2.4.10. *Consider the following procedure:*

i. *Decompose the edge set of $G_{n,3}$ with the method given in Lemma 2.4.1.*
ii. *Select a merging set S.*
iii. *Recognize the $4^n - 1$ type-I cubes that have their origins in S.*
iv. *Replace each type-I cube with a type-II cube.*

The following statements hold:

1. *After completing step i., we have $c_X = c_Y = c_Z = 4^n$, with different components of X being Z_i^n's, different components of Y being X_i^n's, and different components of Z being Y_i^n's.*
2. *The type-I cubes are pairwise disjoint.*
3. *After fixing S in step ii. and the type-I cubes in step iii., throughout step iv.*
   - *For a fixed i, the vertices of Z_i^n remain in the same component of X, the vertices of X_i^n remain in the same component of Y, and the vertices of Y_i^n remain in the same component of Z, and*
   - *Every merge operation reduces c_X, c_Y, and c_Z by 1.*

   In particular, after finishing step iv., we have an H.D. for $G_{n,3}$.

Proof.

1. This was shown in Lemma 2.4.1.
2. Suppose there exist (x_1, x_2, x_3) and (x'_1, x'_2, x'_3) in S such that their corresponding type-I cubes have some vertex in common, so that for some (i_1, i_2, i_3) and (i'_1, i'_2, i'_3) in $\{0,1\}^3$ we have

$$(x_1, x_2, x_3) + (i_1, i_2, i_3) = (x'_1, x'_2, x'_3) + (i'_1, i'_2, i'_3) \pmod{4^n}.$$

   It follows that $x_r + i_r = x'_r + i'_r \pmod{4^n}$ for each $1 \leqslant r \leqslant 3$. Adding these congruences we get $x_1 + x_2 + x_3 + i_1 + i_2 + i_3 = x'_1 + x'_2 + x'_3 + i'_1 + i'_2 + i'_3 \pmod{4^n}$, but $x_1 + x_2 + x_3 = x'_1 + x'_2 + x'_3 = -1 \pmod{4^n}$, so we obtain $i_1 + i_2 + i_3 = i'_1 + i'_2 + i'_3 \pmod{4^n}$, and in particular, $i_1 + i_2 + i_3 = i'_1 + i'_2 + i'_3 \pmod 2$. This implies that $i_r = i'_r$ for

some r, meaning that $x_r = x'_r \pmod{4^n}$ for the same r. This gives $x_r = x'_r$, which contradicts the assumption that S is a merging set.

3. Due to the symmetry involved in (3), it suffices to prove the assertions in just one direction, that is, to prove

   - The vertices of Z_i^n remain in the same component of X, and
   - Every merge operation reduces c_X by 1.

   Without loss of generality, suppose that $S = \{v_0, v_1, \ldots, v_{4^n-2}\}$, where $v_i = (x_i, y_i, i)$, and let L_i be the type-I cube with origin at v_i. Because of (2), the order in which we merge the cubes does not matter, so for the sake of simplicity, assume that L_{4^n-2} is merged first, L_{4^n-3} is merged next, and so on.

   We proceed by induction. The base case is satisfied due to Obsevation 2.4.4 and Lemma 2.4.8. Suppose that we have merged cubes L_{4^n-2} to L_i. The induction hypothesis states that we have cycles $X \cap Z_1^n, X \cap Z_2^n, \ldots, X \cap Z_{i-1}^n$, and a long cycle $X \cap Z_{[i,4^n-1]}^n$. It also states that the vertices of Z_i^n have been in the same component of X together throughout step iv.. By Observation 2.4.7, the order of the vertices in Z_i^n has not changed yet, so merging L_{i-1} will combine $X \cap Z_{[i,4^n-1]}^n$ and $X \cap Z_{i-1}^n$ into a single cycle $X \cap Z_{[i-1,4^n-1]}^n$. It is clear that the vertices of Z_i^n have remained and will remain in the same component of X.

This completes the proof of the lemma. □

2.4.2. Deriving an H.D. for Q_{6n} from an H.D. for Q_{2n}

Combining the Hamilton decompositions for Q_{2n} and $G_{n,3}$ is very similar to the

2-dimensional case. We think of $G_{n,3}$ as a 3-dimensional cyclic grid, and assign coordinates like (x, y, z) to its vertices.

Definition 2.4.11. *Let E be a directed Hamilton cycle in Q_{2n}. A* 3-dimensional seating *of Q_{6n} onto $G_{n,3}$ via E, is a representation of the vertices of Q_{6n} by assigning them integral coordinates as follows:*

1. *Consider E and its positive direction. Assign 0 to $\mathbf{0}$, assign 1 to the next vertex in E, and continue until $4^n - 1$ is assigned to the last vertex of E.*
2. *Induce the order of E onto Q_{2n}, so that each vertex has the same order in either graph.*
3. *Using the coordinates in (2), assign coordinates to every member of $Q_{6n} = Q_{2n} \square Q_{2n} \square Q_{2n}$. Put the vertices on the 3-dimensional grid using their coordinates.*

Using the natural order of E, we have mapped the vertices of Q_{6n} onto $G_{n,3}$ and recognized Q_{6n} as a supergraph of $G_{n,3}$. If H is a directed Hamilton cycle in $G_{n,3}$, the 3-dimensional directed Hamilton cycle derived from E and H, *denoted by $g(E, H)$, is a Hamilton cycle in Q_{6n} and is defined in the natural way:*

1. *3-dimensionally seat Q_{6n} onto $G_{n,3}$ via E.*
2. *Q_{6n} has $3n$ axes $1, 2, \ldots, 3n$, while $G_{n,3}$ has an x-axis, a y-axis, and a z-axis. The axes $1, 2, \ldots, n$ are in direction x, the axes $n+1, n+2, \ldots, 2n$ are in direction y, and the axes $2n+1, 2n+2, \ldots, 3n$ are in direction z.*
3. *$g(E, H)$ has the same edges in the supergraph $Q_{2n} \square Q_{2n} \square Q_{2n}$ as H has in the subgraph $G_{n,3}$.*

Lemma 2.4.12. *Let H_1 and H_2 be two disjoint Hamilton cycles in $G_{n,3}$ and E_1 and E_2 be two disjoint Hamilton cycles in Q_{2n}. Then the four Hamilton cycles $G_1 = g(E_1, H_1)$, $G_2 = g(E_1, H_2)$, $G_3 = g(E_2, H_1)$, and $G_4 = g(E_2, H_2)$ in Q_{6n} are pairwise disjoint.*

Proof. We show that $G_1 = g(E_1, H_1)$ is disjoint from the other three cycles G_2, G_3, and G_4. To achieve this, we 3-dimensionally seat Q_{6n} onto $G_{n,3}$ via E_1. Similarly to the 2-dimensional case, G_1 and G_2 have all their edges on the grid, while G_3 and G_4 have all their edges off the grid. Thus G_1 is disjoint from G_3 and G_4. Furthermore, G_1 and G_2 represent H_1 and H_2, respectively, and H_1 and H_2 are disjoint, so G_1 and G_2 must be disjoint as well. □

Corollary 2.4.13. *If $\{H_1, H_2, H_3\}$ is an H.D. for $G_{n,3}$ and $\{E_1, E_2, \ldots, E_n\}$ is an H.D. for Q_{2n}, then the family $\{g(E_i, H_j) \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 3\}$ is an H.D. for Q_{6n}. The new Hamilton cycles are named F_1, F_2, $\ldots$, F_{3n} via $F_j = f(E_j, H_1)$, $F_{j+n} = f(E_j, H_2)$, and $F_{j+2n} = f(E_j, H_3)$ for $1 \leqslant j \leqslant n$.*

2.4.3. An algorithm for computing an H.D. for $G_{n,3}$

In Section 2.4.1 we saw how to derive a Hamilton decomposition $\{X, Y, Z\}$ from the initial partitioning given by Lemma 2.4.1. We now give an algorithm to compute X. Algorithms for Y and Z are similar.

The idea is to apply the edge decomposition given in Lemma 2.4.1, and then proceed from the origin, initially moving in the positive direction of X, until we reach a chosen type-I cube (one whose origin belongs to the merging set). We then recognize the special vertex, take the necessary actions mandated by the merge operation, and continue to

walk in X. Figure 2.5 shows all the special vertices and the reasoning behind our actions. For example, if we reach m' and the current direction is negative, it means that we came from outside of the cube (and not from m), so we should go to m and change the direction to positive, so that we move outside in the next step. If we reach m' and the current direction is positive, however, it means that we came from m (and not from outside), so we should move outside and leave the direction unchanged.

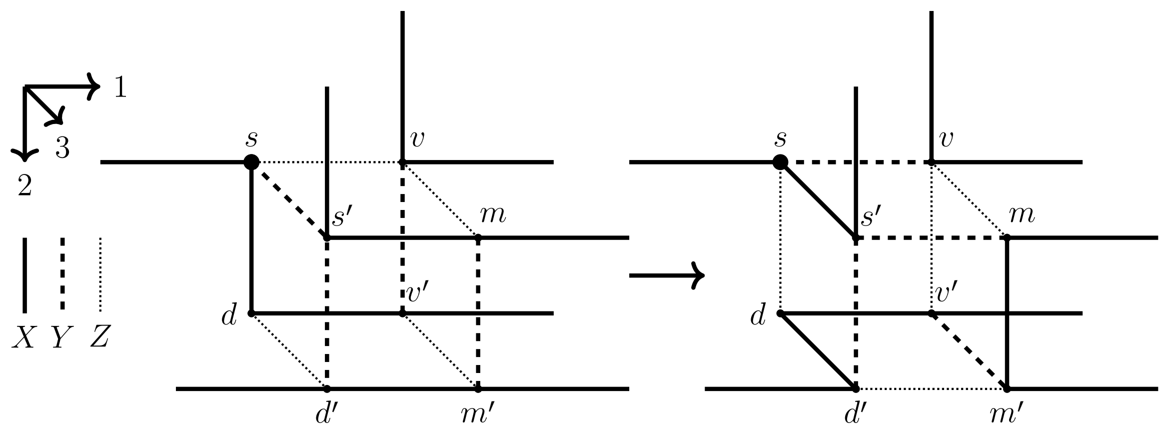


Figure 2.5: A merge operation together with the attached X-edges. The above vertex labelling conforms to that of Algorithm 2.

We choose the merging set to be

$$S = \left\{ (0, 0, 4^t - 1), (1, 1, 4^t - 3), \ldots, \left(\frac{4^t}{2} - 1, \frac{4^t}{2} - 1, 1\right), \right.$$
$$\left. \left(\frac{4^t}{2}, \frac{4^t}{2} + 1, 4^t - 2\right), \left(\frac{4^t}{2} + 1, \frac{4^t}{2} + 2, 4^t - 4\right), \ldots, (4^t - 2, 4^t - 1, 2) \right\}.$$

We choose S like this for two reasons:

- The origin belongs to none of the type-I cubes, so we do not need an initial case check.
- The set S has all the x-coordinates from 0 to $4^n - 2$, so it is easy to check if a coordinate belongs to it.

We define five auxiliary sets S', D, D', M, and M' so that we have immediate access to all the special vertices. Algorithm 2 given in Section 2.7 calculates X.

2.4.4. 3-Dimensional Algorithm

Just like in the 2-dimensional case, we use the definition of $g(E,H)$ to devise an algorithm for computing an H.D. for Q_{6n}. Algorithm 3 is very similar to Algorithm 1, and is given in Section 2.7.

2.5. Highly Symmetric Hamilton Decompositions

The theory we have developed in the previous chapters can be improved to give us highly symmetric Hamilton decompositions. Let $\sigma : [1,k] \to [1,k]$ be a permutation. Then σ induces a homomorphism of $G_{n,k}$ by relabelling the axes: The axis previously referred to as i is now called $\sigma(i)$. More specifically, the vertex $v = (x_1, x_2, \ldots, v_k)$ is mapped to $\sigma(v) = (x_{\sigma^{-1}(1)}, x_{\sigma^{-1}(2)}, \ldots, x_{\sigma^{-1}(k)})$. As σ is a homomorphism, it maps Hamilton cycles to Hamilton cycles. If $H = e_1 e_2 \ldots e_{4^{nk}}$ is a directed Hamilton cycle in $G_{n,k}$, then $\sigma(H)$ is the Hamilton cycle

$$\sigma(e_1)\sigma(e_2)\ldots\sigma(e_{4^{nk}})$$

Note that σ maps backward edges to backward edges: If $\sigma(i) = j$, then $\sigma(\bar{i}) = \bar{j}$. It is worth remembering that $\bar{i}$ stands for an edge from $(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ to $(x_1, x_2, \ldots, x_{i-1}, x_i - 1 \pmod{4^n}, x_{i+1}, \ldots, x_n)$.

Definition 2.5.1. *A family $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ of k permutations on $[1,k]$ is called a* Latin family *if the matrix $m_{ij} = \sigma_i(j)$ is a Latin square. We do not differentiate between σ_i*

and the i*th row of the matrix. For the sake of simplicity, we require that* σ_1*, the first row of the matrix, is the identity.*

Let $T = \{H_1, H_2, \dots, H_k\}$ *be an H.D. for* $G_{n,k}$*. We say that* T *is a* Latin Hamilton decomposition *if there exists a Hamilton cycle* H *in* $G_{n,k}$ *and a Latin family* $\mathcal{S} = \{\sigma_1, \sigma_2, \dots, \sigma_k\}$ *of permutations on* $[1, k]$ *such that*

$$H_i = \sigma_i(H) \qquad \textit{for all} \qquad 1 \leqslant i \leqslant k.$$

The Hamilton cycle H *(=* H_1*) is then called a* source cycle *for* $G_{n,k}$ *and the matrix* $m_{ij} = \sigma_i(j)$ *is called a* source matrix *for* $G_{n,k}$*. The pair* (H, M) *is called a* source pair *for* $G_{n,k}$*.*

The H.D. given for $G_{n,2}$ in 2.3.1 is Latin, but the one given for $G_{n,3}$ in 2.4.1 is not necessarily so. If it is not Latin, we can turn it into one with a small adjustment.

Theorem 2.5.2. *The set* S *mentioned in Lemma 2.4.10 step ii. can be chosen in such a way that the resulting H.D. is Latin. More specifically, if*

$$\begin{aligned} S^* = \Bigg\{ &\left(0, \frac{4^t}{2} - 1, \frac{4^t}{2}\right), \left(1, \frac{4^t}{2} - 3, \frac{4^t}{2} + 1\right), \dots, \left(\frac{4^t - 4}{6}, \frac{4^t + 2}{6}, \frac{4 \times 4^t - 4}{6}\right), \\ &\left(\frac{4 \times 4^t + 2}{6}, \frac{5 \times 4^t + 4}{6}, \frac{4^t}{2} - 2\right), \left(\frac{4 \times 4^t + 2}{6} + 1, \frac{5 \times 4^t + 4}{6} + 1, \frac{4^t}{2} - 4\right), \dots, \\ &\left(\frac{5 \times 4^t - 2}{6} - 1, 4^t - 1, \frac{4^t + 2}{6} + 1\right) \Bigg\} \end{aligned}$$

and

$$S = \left\{(x, y, z) \;\middle|\; (x, y, z) \in S^*, \ \textit{or} \ (y, z, x) \in S^*, \ \textit{or} \ (z, x, y) \in S^* \right\}, \tag{2.5.1}$$

then the resulting H.D. is Latin.

Proof. We show that it suffices for S to have the following property:

If $(x, y, z) \in S$, then $(y, z, x) \in S$ and $(z, x, y) \in S$.

To see this, consider $G_{n,3}$ after completion of Lemma 2.4.10 step i.. Let $\sigma_i : [1,3] \to [1,3]$ be defined via $\sigma_i(j) = i + j - 1 \pmod 3$ for i and j in $[1,3]$. It is not hard to see that

$$\sigma_2(X) = Y \text{ and } \sigma_3(X) = Z. \tag{2.5.2}$$

We wish to show that the relations given in 2.5.2 remain valid after completion of Lemma 2.4.10 step iv.. To achieve this, we merge the cubes three at a time and use induction.

Suppose that $u_1 = (x,y,z)$, $u_2 = \sigma_2(u_1) = (z,x,y)$, and $u_3 = \sigma_3(u_1) = (y,z,x)$ belong to S, and let L_1, L_2, and L_3 be type-I cubes with their origins at u_1, u_2, and u_3, respectively. By the induction hypothesis, we know that 2.5.2 is valid before merging L_1, L_2, and L_3.

Since $u_2 = \sigma_2(u_1)$, we have $L_2 = \sigma_2(L_1)$, and because $u_3 = \sigma_3(u_1)$, we get $L_3 = \sigma_3(L_1)$. Furthermore, analyzing the merge operator gives $\sigma_2(L_1 \cap X) = L_2 \cap Y$ and $\sigma_3(L_1 \cap X) = L_3 \cap Z$. This means that 2.5.2 is valid after merging the three cubes. Therefore X (after finishing Lemma 2.4.10 step iv.) is a source cycle for $G_{n,3}$ in the H.D. $\{X,Y,Z\}$, and its source matrix is $\begin{bmatrix} \begin{array}{c|c|c} 1 & 2 & 3 \\ \hline 2 & 3 & 1 \\ \hline 3 & 1 & 2 \end{array} \end{bmatrix}$. $\square$

We may modify Algorithm 1 to take source pairs for Q_{2n} and $G_{n,2}$ and produce a source pair for Q_{4n}. We may also modify Algorithm 3 to take source pairs for Q_{2n} and $G_{n,3}$ and produce a source pair for Q_{6n}. Algorithms 4 and 5 are the Latin counterparts to Algorithms 1 and 3, respectively, and are given in Section 2.7. We may also specify that Algorithm 2 takes a suitable merging set (2.5.1) so that it produces a source cycle for $G_{n,3}$. Hence, it is not necessary to give a Latin counterpart to Algorithm 2.

Theorem 2.5.3. *If $\{H_1, H_2\}$ and $\{E_1, E_2, \ldots, E_n\}$ mentioned in Corollary 2.3.3 are Latin,*

then the resulting Hamilton decomposition $\{f(E_i, H_j) \mid 1 \leqslant i \leqslant n$ *and* $1 \leqslant j \leqslant 2\}$ *is also Latin.*

Proof. Let E_1, our source cycle for Q_{2n}, have source matrix M. For $G_{n,2}$, the cycle H_1 is a source cycle and has source matrix $\left[\begin{array}{c|c} 1 & 2 \\ \hline 2 & 1 \end{array}\right]$. We show that $F_1 = f(E_1, H_1)$ is a source cycle for Q_{4n} with $M' = \left[\begin{array}{c|c} M & M+n \\ \hline M+n & M \end{array}\right]$ as its source matrix, where $M + n$ is obtained from M by adding n to every entry.

Our proof is based on Algorithm 1. In Section 2.8 it is shown that Algorithm 1 computes $f(E, H)$ correctly. We know that, for $1 \leqslant j \leqslant n$, this algorithm stores $f(E_j, H_1)$ and $f(E_j, H_2)$ as F_j and F_{j+n}, respectively. The dimension of the ith edge of F_j is stored in $f[j-1][i-1][0]$ and its direction is stored in $f[j-1][i-1][1]$. Due to line 9 in the algorithm and the fact that H_1 and H_2 make a Latin decomposition, for every $1 \leqslant i \leqslant 4^{2n}$, either all the F_i's have a forward edge in the ith position or all the F_i's have a backward edge in the ith position. So the directions of the edges are as required and we only need to focus on their dimensions.

To show that the edge dimensions are as we want, we define a $2n$ by $2n$ matrix Q via the following:

$q_{i,j} = t$ if there is some $0 \leqslant s < 4^{2n}$ such that $f[0][s][0] = j$ and $f[i+1][s][0] = t$.

We show that

- Q is well-defined, and
- $Q = M'$.

This would complete the proof of the theorem.

For $1 \leqslant i \leqslant 2n$, let S_i be the set of edge numbers in F_1 with dimension i. More

precisely,

$$S_i = \{j \mid 0 \leqslant j < 4^{2n} \text{ and } f[0][j][0] = i\}$$

Suppose that $1 \leqslant v \leqslant n$ and let $s \in S_v$. Lines 10, 12, and 23 say that, for $j = 0$, $i = s$, and $k = 0$, we have $\dim = 0$ and that for $u = c[k][\dim]$ we have $f[0][s][0] = e[0][u][0]$, but $s \in S_v$, so we have $f[0][s][0] = e[0][u][0] = v = m'_{1,v} = m_{1,v}$. Therefore, $q_{1,v}$ is well-defined and is equal to $m_{1,v}$. Again, due to lines 10, 12, and 23, for $0 \leqslant w < n$, putting $j = w$ but keeping the same i and k, we have the same u, and thus $f[w][s][0] = e[w][u][0] = m_{w+1,v}$. This means that $q_{w+1,v}$ is well-defined as is equal to $m_{w+1,v}$. Since w and v were arbitrary in $[0, n-1]$ and $[1, n]$, respectively, we get $q_{w+1,v} = m_{w+1,v}$ for $1 \leqslant v \leqslant n$ and $0 \leqslant w < n$.

A similar argument for the other cases shows that

- for $n+1 \leqslant v \leqslant 2n$ and $1 \leqslant w \leqslant n$ we have $q_{w,v} = m_{w,v-n} + n$,
- for $1 \leqslant v \leqslant n$ and $n+1 \leqslant w \leqslant 2n$ we have $q_{w,v} = m_{w-n,v} + n$, and
- for $n+1 \leqslant v \leqslant 2n$ and $n+1 \leqslant w \leqslant 2n$ we have $q_{w,v} = m_{w-n,v-n}$.

This shows that Q is well-defined and $Q = M'$. □

As a corollary, we have the following important result.

Corollary 2.5.4. *If Q_{2n} has a source cycle, so does Q_{4n}.*

Theorem 2.5.5. *If $\{H_1, H_2, H_3\}$ and $\{E_1, E_2, \ldots, E_n\}$ mentioned in Corollary 2.4.13 are Latin, then the resulting Hamilton decomposition $\{g(E_i, H_j) \mid 1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant 3\}$ is also Latin.*

Proof. The proof is very similar to that of Theorem 2.5.3, therefore we only sketch it here. Based on Algorithm 3, if E_1 is a source cycle for Q_{2n} with source matrix M, and if H_1 is

a source cycle for $G_{n,3}$ with source matrix $\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$, then $g(E_1, H_1)$ is a source cycle for Q_{6n} with $M' = \begin{bmatrix} M & M+n & M+2n \\ M+n & M+2n & M \\ M+2n & M & M+n \end{bmatrix}$ as its source matrix. $\square$

The last theorem gives rise to another important result:

Corollary 2.5.6. *If Q_{2n} has a source cycle, so does Q_{6n}.*

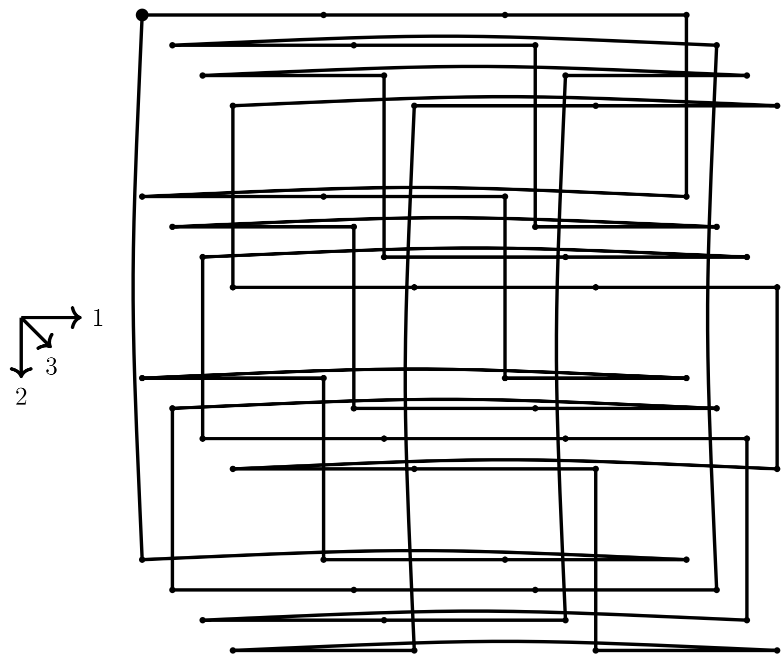
Corollaries 2.5.4 and 2.5.6 give us the main result of this chapter:

Corollary 2.5.7. *We have a source cycle for all Q_{2n} with $n = 2^a 3^b$.*

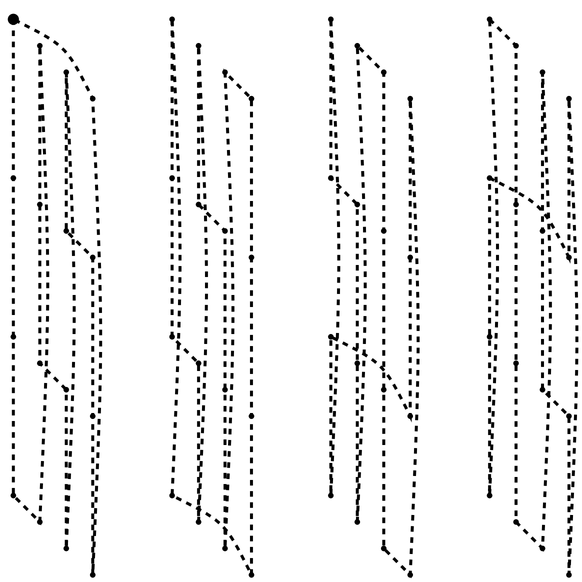
For future research, we conjecture the following.

Conjecture 2.5.8. *We have a source cycle for all Q_{2n}.*

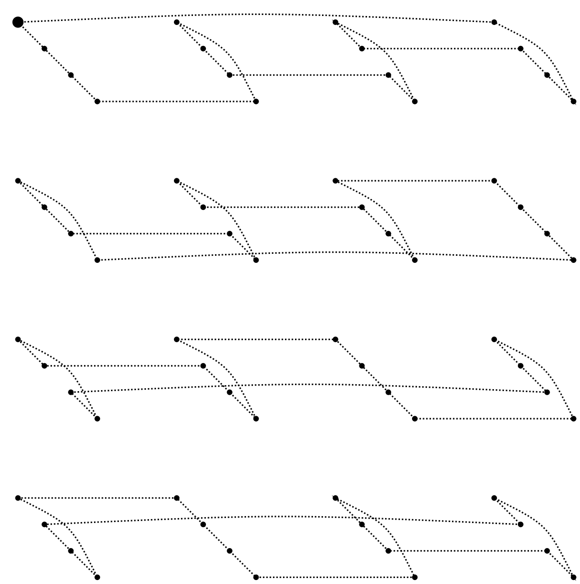
2.6. Lemma 2.4.1 for $n = 1$



(a) X



(b) Y



(c) Z

Figure 2.6: The decomposition discussed in Lemma 2.4.1 for $n = 1$.

2.7. Algorithms

2.7.1. An H.D. for Q_{4n}

Input:

- An $n \times 4^n$ array e with its ith row showing the ith Hamilton cycle of Q_{2n}.
- A 2×4^{2n} array h with its ith row showing the ith Hamilton cycle of $G_{n,2}$.

Output:

- A $2n \times 4^{2n}$ array f with its ith row showing the ith Hamilton cycle of Q_{4n}.

Algorithm 1 An H.D. for Q_{4n} from an H.D. for Q_{2n} and an H.D. for $G_{n,2}$

```
for i ← 0 to 1 do
    for j ← 0 to 1 do
        c[i][j] ← 0                  ▷ initializing the x- and y-coordinates of the two pointers
    end for
end for
for j ← 0 to n − 1 do                                          ▷ cycling through E_1 to E_n
    for i ← 0 to 4^{2n} − 1 do                         ▷ cycling through edges of H_1 and H_2
        for k ← 0 to 1 do                                      ▷ cycling through H_1 and H_2
            dir ← h[k][i][1]                     ▷ direction of the current edge in H_{k+1}
            dim ← h[k][i][0] − 1                 ▷ dimension of the current edge in H_{k+1}
            if dir = 0 then                      ▷ if the current edge in H_{k+1} is forward
                f[j + kn][i][0] ← e[j][c[k][dim]][0] + n(dim)   ▷ dimension of the current edge in F_{j+1+kn}
                f[j + kn][i][1] ← e[j][c[k][dim]][1]            ▷ direction of the current edge in F_{j+1+kn}
                c[k][dim] ← c[k][dim] + 1       ▷ moving forward in the current copy of E_{j+1}
                if c[k][dim] = 4^n then                                    ▷ mod operations
                    c[k][dim] ← 0
                end if
            else                                ▷ if the current edge in H_{k+1} is backward
                c[k][dim] ← c[k][dim] − 1       ▷ moving backward in the current copy of E_{j+1}
                if c[k][dim] = −1 then                                     ▷ mod operations
                    c[k][dim] ← 4^n − 1
                end if
                f[j + kn][i][0] ← e[j][c[k][dim]][0] + n(dim)   ▷ dimension of the current edge in F_{j+1+kn}
                f[j + kn][i][1] ← 1 − e[j][c[k][dim]][1]        ▷ direction of the current edge in F_{j+1+kn}
            end if
        end for
    end for
end for
```

2.7.2. An H.D. for $G_{n,3}$

Input:

- A $4^n \times 2$ array $S[][]$ having the merging set S in its first $4^n - 1$ rows.
  The elements of S are sorted by their x-coordinates, with the ith row of $S[][]$ having the element with x-cooridnate i. The first entry gives the y-coordinate and the second gives the z-coordinate.

Output:

- A $4^{3n} \times 2$ array H having the edges of X.

Algorithm 2 An algorithm for finding X.

```
x ← 0                                   ▷ initializing the pointer's x-coordinate
y ← 0                                   ▷ initializing the pointer's y-coordinate
z ← 0                                   ▷ initializing the pointer's z-coordinate
c ← 0                                   ▷ c = x + y + z
dir ← 0
s[4^n − 1][0] ← −2                      ▷ no element of S has x-coordinate equal to 4^n − 1
s[4^n − 1][1] ← −2
for i ← 0 to 4^n − 1 do                 ▷ creating the auxiliary sets S′, D, D′, M, and M′
    sp[i][0] ← s[i][0]                  ▷ creating the ith member of S′
    sp[i][1] ← s[i][1] + 1
    if sp[i][1] = 4^n then              ▷ mod operations
        sp[i][1] ← 0
    end if
    d[i][0] ← s[i][0] + 1               ▷ creating the ith member of D
    d[i][1] ← s[i][1]
    if d[i][0] = 4^n then               ▷ mod operations
        d[i][0] ← 0
    end if
    dp[i][0] ← d[i][0]                  ▷ creating the ith member of D′
    dp[i][1] ← sp[i][1]
    m[i + 1][0] ← s[i][0]               ▷ creating the ith member of M
    m[i + 1][1] ← sp[i][1]
    mp[i + 1][0] ← d[i][0]              ▷ creating the ith member of M′
    mp[i + 1][1] ← sp[i][1]
end for
m[0][0] ← −1                            ▷ no element of M has x-coordinate equal to 0
m[0][1] ← −1
mp[0][0] ← −1                           ▷ no element of M′ has x-coordinate equal to 0
mp[0][1] ← −1
```

```
for i ← 0 to 4^{3n} − 1 do                          ▷ main loop for building the ith edge
    if s[x][0] = y and s[x][1] = z then                                  ▷ (x, y, z) ∈ S
        if dir = 0 then                          ▷ we have reached S from outside of cube
            dir ← 1               ▷ in the next step we exit from S′ in negative direction
            h[i][0] ← 3                                 ▷ the ith edge is in dimension 3
            h[i][1] ← 0                           ▷ the ith edge is in positive direction
            c ← c + 1                                            ▷ adding 1 to z and c
            z ← z + 1
            if z = 4^n then                                          ▷ mod operations
                z ← 0
            end if
        else                                              ▷ we have reached S from S′
            dir ← 1
            h[i][0] ← 1                                 ▷ the ith edge is in dimension 1
            h[i][1] ← 1                           ▷ the ith edge is in negative direction
            c ← c − 1                                     ▷ subtracting 1 from x and c
            x ← x − 1
            if x = −1 then                                           ▷ mod operations
                x ← 4^n − 1
            end if
        end if
    else if sp[x][0] = y and sp[x][1] = z then                          ▷ (x, y, z) ∈ S′
        if dir = 0 then                         ▷ we have reached S′ from outside of cube
            dir ← 1                ▷ in the next step we exit from S in negative direction
            h[i][0] ← 3                                 ▷ the ith edge is in dimension 3
            h[i][1] ← 1                           ▷ the ith edge is in negatice direction
            c ← c − 1                                     ▷ subtracting 1 from z and c
            z ← z − 1
            if z = −1 then                                           ▷ mod operations
                z ← 4^n − 1
            end if
        else                                              ▷ we have reached S′ from S
            dir ← 1
            h[i][0] ← 2                                 ▷ the ith edge is in dimension 2
            h[i][1] ← 1                           ▷ the ith edge is in negative direction
            c ← c − 1                                     ▷ subtracting 1 from z and c
            y ← y − 1
            if y = −1 then                                           ▷ mod operations
                y ← 4^n − 1
            end if
        end if
```

```
    else if d[x][0] = y and d[x][1] = z then            ▷ (x, y, z) ∈ D
        if dir = 0 then                                 ▷ we have reached D from D′
            h[i][0] ← 1                                 ▷ the ith edge is in dimension 1
            h[i][1] ← 0                                 ▷ the ith edge is in positive direction
            c ← c + 1                                   ▷ adding 1 to x and c
            x ← x + 1
            if x = 4^n then                             ▷ mod operations
                x ← 0
            end if
        else                                            ▷ we have reached D from V′
            h[i][0] ← 3                                 ▷ the ith edge is in dimension 3
            h[i][1] ← 0                                 ▷ the ith edge is in positive direction
            c ← c + 1                                   ▷ adding 1 to x and c
            z ← z + 1
            if z = 4^n then                             ▷ mod operations
                z ← 0
            end if
        end if
    else if dp[x][0] = y and dp[x][1] = z then          ▷ (x, y, z) ∈ D′
        if dir = 0 then                                 ▷ we have reached D′ from outside of cube
            h[i][0] ← 3                                 ▷ the ith edge is in dimension 3
            h[i][1] ← 1                                 ▷ the ith edge is in negative direction
            c ← c − 1                                   ▷ subtracting 1 from z and c
            z ← z − 1
            if z = −1 then                              ▷ mod operations
                z ← 4^n − 1
            end if
        else                                            ▷ we have reached D′ from D
            h[i][0] ← 1                                 ▷ the ith edge is in dimension 1
             h[i][1] ← 1                                ▷ the ith edge is in negative direction
             c ← c − 1                                  ▷ subtracting 1 from z and c
             x ← x − 1
             if x = −1 then                             ▷ mod operations
                 x ← 4^n − 1
             end if
         end if
```

```
    else if m[x][0] = y and m[x][1] = z then                    ▷ (x, y, z) ∈ M
        if dir = 0 then                                         ▷ we have reached M from M′
            h[i][0] ← 1                                         ▷ the ith edge is in dimension 1
            h[i][1] ← 0                                         ▷ the ith edge is in positive direction
            c ← c + 1                                           ▷ adding 1 to x and c
            x ← x + 1
            if x = 4^n then                                     ▷ mod operations
                x ← 0
            end if
        else                                                    ▷ we have reached M from outside of cube
            dir ← 0                                             ▷ in the next step we exit from M′ in positive direction
            h[i][0] ← 2                                         ▷ the ith edge is in dimension 3
            h[i][1] ← 0                                         ▷ the ith edge is in positive direction
            c ← c + 1                                           ▷ adding 1 to y and c
            y ← y + 1
            if y = 4^n then                                     ▷ mod operations
                y ← 0
            end if
        end if
    else if mp[x][0] = y and mp[x][1] = z then                  ▷ (x, y, z) ∈ M′
        if dir = 0 then                                         ▷ we have reached M′ from M
            h[i][0] ← 1                                         ▷ the ith edge is in dimension 1
            h[i][1] ← 0                                         ▷ the ith edge is in positive direction
            c ← c + 1                                           ▷ adding 1 to x and c
            x ← x + 1
            if x = 4^n then                                     ▷ mod operations
                x ← 0
            end if
        else                                                    ▷ we have reached M′ from outside of cube
            dir ← 1                                             ▷ in the next step we exit from M in positive direction
            h[i][0] ← 2                                         ▷ the ith edge is in dimension 2
            h[i][1] ← 1                                         ▷ the ith edge is in negative direction
            c ← c − 1                                           ▷ subtracting 1 from y and c
            y ← y − 1
            if y = −1 then                                      ▷ mod operations
                y ← 4^n − 1
            end if
        end if
```

```
    else                                          ▷ normal vertex
        if dir = 0 then                           ▷ if the current direction is positive
            if c = −1 (mod 4^n) then              ▷ if it is time to move in dimension 2
                h[i][0] ← 2                       ▷ the ith edge is in dimension 2
                h[i][1] ← 0                       ▷ the ith edge is in the positive direction
                c ← c + 1                         ▷ adding 1 to y and c
                y ← y + 1
                if y = 4^n then                   ▷ mod operations
                    y ← 0
                end if
            else                                  ▷ if it is time to move in dimension 1
                h[i][0] ← 1                       ▷ the ith edge is in dimension 1
                h[i][1] ← 0                       ▷ the ith edge is in positive direction
                c ← c + 1                         ▷ adding 1 to x and c
                x ← x + 1
                if x = 4^n then                   ▷ mod operations
                    x ← 0
                end if
            end if
        else                                      ▷ if the current direction is negative
            if c = 0 (mod 4^n) then               ▷ if it is time to move in dimension 2
                h[i][0] ← 2                       ▷ the ith edge is in dimension 2
                h[i][1] ← 1                       ▷ the ith edge is in negative direction
                c ← c − 1                         ▷ subtracting 1 from y and c
                y ← y − 1
                if y = −1 then                    ▷ mod operations
                    y ← 4^n − 1
                end if
            else                                  ▷ if it is time to move in dimension 1
                h[i][0] ← 1                       ▷ the ith edge is in dimension 1
                h[i][1] ← 1                       ▷ the ith edge is in negative direction
                c ← c − 1                         ▷ subtracting 1 from x and c
                x ← x − 1
                if x = −1 then                    ▷ mod operations
                    x ← 4^n − 1
                end if
            end if
        end if
    end if
end for
```

2.7.3. An H.D. for Q_{6n}

Input:

- An $n \times 4^n$ array e with its ith row showing the ith Hamilton cycle for Q_{2n}.
- A 3×4^{3n} array h with its ith row showing the ith Hamilton cycle for $G_{n,3}$.

Output:

- A $3n \times 4^{3n}$ array g with its ith row showing the ith Hamilton cycle for Q_{6n}.

Algorithm 3 An H.D. for Q_{6n} from an H.D. for Q_{2n} and an H.D. for $G_{n,3}$

```
for i ← 0 to 2 do
    for j ← 0 to 2 do
        c[i][j] ← 0        ▷ initializing the x-, y-, and z-coordinates of the three pointers
    end for
end for
for j ← 0 to n − 1 do                          ▷ cycling through E_1 to E_n
    for i ← 0 to 4^{3n} − 1 do                 ▷ cycling through edges of H_1, H_2, and H_3
        for k ← 0 to 2 do                      ▷ cycling through H_1, H_2, and H_3
            dir ← h[k][i][1]                   ▷ direction of the current edge in H_{k+1}
            dim ← h[k][i][0] − 1               ▷ dimension of the current edge in H_{k+1}
            if dir = 0 then                    ▷ if the current edge in H_{k+1} is forward
                f[j + kn][i][0] ← e[j][c[k][dim]][0] + n(dim)   ▷ dimension of the current edge in F_{j+1+kn}
                f[j + kn][i][1] ← e[j][c[k][dim]][1]            ▷ direction of the current edge in F_{j+1+kn}
                c[k][dim] ← c[k][dim] + 1      ▷ moving forward in the current copy of E_{j+1}
                if c[k][dim] = 4^n then        ▷ mod operations
                    c[k][dim] ← 0
                end if
            else                               ▷ if the current edge in H_{k+1} is backward
                c[k][dim] ← c[k][dim] − 1      ▷ moving backward in the current copy of E_{j+1}
                if c[k][dim] = −1 then         ▷ mod operations
                    c[k][dim] ← 4^n − 1
                end if
                f[j + kn][i][0] ← e[j][c[k][dim]][0] + n(dim)   ▷ dimension of the current edge in F_{j+1+kn}
                f[j + kn][i][1] ← 1 − e[j][c[k][dim]][1]        ▷ direction of the current edge in F_{j+1+kn}
            end if
        end for
    end for
end for
```

2.7.4. A source cycle for Q_{4n}

Input:

- A $4^n \times 2$ array e having the source cycle for Q_{2n}.
- An n by n source matrix A for the cycle E.
- A $4^{2n} \times 2$ array h having the source cycle for $G_{n,2}$.

Output:

- A $4^{2n} \times 2$ array f having the source cycle for Q_{4n}.
- A $2n$ by $2n$ matrix P as the accompanying source matrix.

Algorithm 4 A Source Cycle for Q_{4n} From Source Cycles for Q_{2n} and $G_{n,2}$

```
for i ← 0 to n − 1 do                                ▷ building P
    for j ← 0 to n − 1 do
        for k ← 0 to 1 do
            for t ← 0 to 1 do
                z ← (k + t) (mod 2)
                p[i + k][j + tn] ← a[i][j] + zn
            end for
        end for
    end for
end for
for i ← 0 to 1 do            ▷ initializing the x- and y-coordinates of the pointer
    c[i] ← 0
end for
for i ← 0 to 4^(2n) − 1 do                           ▷ cycling through edges of H
    dir ← h[i][1]                                    ▷ direction of the current edge in H
    dim ← h[i][0] − 1                                ▷ dimension of the current edge in H
    if dir = 0 then                                  ▷ if the current edge in H is forward
        f[i][0] ← e[c[dim]][0] + n(dim)              ▷ dimension of the current edge in F
        f[i][1] ← e[c[dim]][1]                       ▷ direction of the current edge in F
        c[dim] ← c[dim] + 1                          ▷ moving forward in the current copy of E
        if c[dim] = 4^n then                         ▷ mod operations
            c[dim] ← 0
        end if
    else                                             ▷ if the current edge in H is backward
        c[dim] ← c[dim] − 1                          ▷ moving backward in the current copy of E
        if c[dim] = −1 then                          ▷ mod operations
            c[dim] ← 4^n − 1
        end if
        f[i][0] ← e[c[dim]][0] + n(dim)              ▷ dimension of the current edge in F
        f[i][1] ← 1 − e[c[dim]][1]                   ▷ direction of the current edge in F
    end if
end for
```

2.7.5. A source cycle for Q_{6n}

Input:

- A $4^n \times 2$ array e having the source cycle for Q_{2n}.
- An n by n source matrix A for the cycle E.
- A $4^{2n} \times 2$ array h having the source cycle for $G_{n,2}$.

Output:

- A $4^{3n} \times 2$ array f having the source cycle for Q_{6n}.
- A $3n$ by $3n$ matrix P as the accompanying source matrix.

Algorithm 5 A Source Cycle for Q_{6n} From Source Cycles for Q_{2n} and $G_{n,3}$

```
for i ← 0 to n − 1 do                                  ▷ building P
    for j ← 0 to n − 1 do
        for k ← 0 to 2 do
            for t ← 0 to 2 do
                z ← (k + t) (mod 3)
                p[i + k][j + tn] ← a[i][j] + zn
            end for
        end for
    end for
end for
for i ← 0 to 2 do          ▷ initializing the x-, y-, and z-coordinates of the pointer
    c[i] ← 0
end for
for i ← 0 to 4^{3n} − 1 do                             ▷ cycling through edges of H
    dir ← h[i][1]                                      ▷ direction of the current edge in H
    dim ← h[i][0] − 1                                  ▷ dimension of the current edge in H
    if dir = 0 then                                    ▷ if the current edge in H is forward
        f[i][0] ← e[c[dim]][0] + n(dim)                ▷ dimension of the current edge in F
        f[i][1] ← e[c[dim]][1]                         ▷ direction of the current edge in F
        c[dim] ← c[dim] + 1                            ▷ moving forward in the current copy of E
        if c[dim] = 4^n then                           ▷ mod operations
            c[dim] ← 0
        end if
    else                                               ▷ if the current edge in H is backward
        c[dim] ← c[dim] − 1                            ▷ moving backward in the current copy of E
        if c[dim] = −1 then                            ▷ mod operations
            c[dim] ← 4^n − 1
        end if
        f[i][0] ← e[c[dim]][0] + n(dim)                ▷ dimension of the current edge in F
        f[i][1] ← 1 − e[c[dim]][1]                     ▷ direction of the current edge in F
    end if
end for
```

2.8. Correctness of Algorithm 1

For a fixed j, in the i-loop, Algorithm 1 outputs two cycles $F_{j+1} = f(E_{j+1}, H_1)$ and $F_{j+1+n} = f(E_{j+1}, H_2)$. It does so by traversing the edges of H_1 ($k = 0$) and H_2 ($k = 1$) and mimicking them:

- If the ith edge of H_1 is 1 or $\overline{1}$, then the ith edge of F_{j+1} is one of $\{1, \overline{1}, 2, \overline{2}, \ldots, n, \overline{n}\}$, and if the ith edge of H_1 is 2 or $\overline{2}$, then the ith edge of F_{j+1} is one of $\{n+1, \overline{n+1}, n+2, \overline{n+2}, \ldots, 2n, \overline{2n}\}$. Same thing is true for H_2 and F_{j+1+n}.
- A pointer, with its x- and y-coordinates being $c[0][0]$ and $c[0][1]$, tracks movement through H_1 on the 2-dimensional grid. Another pointer, with its x- and y-coordinates being $c[1][0]$ and $c[1][1]$, tracks movement through H_2 on the 2-dimensional grid. These pointers together with E_{j+1} determine in what dimension and direction the ith edges of F_{j+1} F_{j+1+n} are:
  - If the ith edge of H_1 is from (a, b) to $(a+1, b)$, then the ith edge of F_{j+1} has the same direction and dimension as the $(a+1)st$ edge of E_{j+1}.
  - If the ith edge of H_1 is from (a, b) to $(a, b+1)$, then the ith edge of F_{j+1} has direction equal to that of the $(b+1)st$ edge of E_{j+1} and dimension equal to n plus the dimension of the $(b+1)st$ edge of E_{j+1}.
  - If the ith edge of H_1 is from (a, b) to $(a-1, b)$, then the ith edge of F_{j+1} has direction opposite to that of the ath edge of E_{j+1} and dimension equal to that of the ath edge of E_{j+1}.
  - If the ith edge of H_1 is from (a, b) to $(a, b-1)$, then the ith edge of F_{j+1} has

direction opposite to that of the *bth* edge of E_{j+1} and dimension equal to n plus the dimension of the *bth* edge of E_{j+1}.

  - Similar statements can be made about H_2 and E_{j+1}.

- The pointers are initially set to $(0, 0)$. After each iteration of j, the pointers become $(0, 0)$ because H_1 and H_2 start and end at $(0, 0)$.

Chapter 3. Contraction Decomposition

3.1. Introduction

In this chapter, all graphs are undirected, but may not be simple, that is, they may have loops and parallel edges. We may emphasize this fact by stating that a graph is a *multigraph*. Let e be an edge of a graph G. The *contraction* of e in G, denoted G/e, is given by identifying the endpoints of e and then deleting e itself. We can see that when e is a loop, contracting e is the same as deleting it. A graph H is a *minor* of G if it can be obtained from G by a series of vertex deletions, edge deletions, and edge contractions. In this case, we write $H \leqslant_m G$. Let u and v be two distinct vertices of a graph G, and let P and Q be two (u,v)-paths. The paths P and Q are called *internally-disjoint* if $V(P) \cap V(Q) = \{v,u\}$. For a graph G on vertex set V, a *partition* of V into k subsets is a grouping of elements of V into k disjoint non-empty subsets. The number k is the *size* of the partition. A partition of the edge set of G is defined analogously. A graph G is called (H,k)-*positive* if its edge set can be partitioned into $E(G) = E_1 \uplus E_2 \uplus \cdots \uplus E_k$ such that $G/(E \setminus E_i)$ does not contain H as a minor for $i \in \{1,2,\ldots,k\}$. If G is not (H,k)-positive, then it is (H,k)-*negative*. A k-*tree* is formed by starting with the complete graph K_{k+1} and then repeatedly doing the following: Recognize a subgraph isomorphic to K_k and add a new vertex adjacent exactly to the vertices of the K_k subgraph. A graph has *treewidth* at most k if it is a subgraph of a k-tree. Throughout this chapter, a graph is said to be the *smallest* among a set of graphs, if it has the fewest number of edges. By the *intersection* of two graphs, we mean the intersection of their edge sets, treated as a graph.

3.2. History

In 1971, Chartrand, Geller, and Hedetniemi asked, among many other questions, whether every planar graph is the union of two outerplanar graphs [7]. In 1996, Ding, Oporowski, Sanders, and Vertigan [10] (and, independently, Kedlaya [14]) showed that every planar graph is the union of two series-parallel graphs. In 2005, Gonçalves showed the conjecture of Chartrand, Geller, and Hedetniemi to be true [12]. Continuing in the same spirit, James Oxley asked the matroid theory question of whether the ground set of every cographic matroid may be partitioned into two sets such that the deletion of either set results in a series-parallel matroid [16]. Translating from matroids to graphs, we get the following, which is the main topic of this chapter [16]:

Conjecture 3.2.1 (Morgan, Oporowski)**.** *Every graph is* $(K_4, 2)$*-positive.*

This conjecture is currently open, but some partial results are known. A result by Demaine, Hajiaghayi, and Mohar [9] guarantees the existence of a 2 edge coloring for graphs of bounded Euler genus such that, contracting each color set, the resulting graph has bounded treewidth. The result of Gonçalves [12] settles Conjecture 3.2.1 for planar graphs, while Theorem 3.3.2, stated in the next section, proves the conjecture for 4-edge-connected graphs. We propose and deal with two slightly different questions derived from changing the parameters in Conjecture 3.2.1.

3.3. First Question

It is clear that the condition of a graph being $(K_4, 3)$-positive is weaker than being $(K_4, 2)$-positive. Is it true that every graph is $(K_4, 3)$-positive? The answer to this question is affirmative.

Theorem 3.3.1. *Every graph is* $(K_4, 3)$*-positive.*

In order to prove this assertion we need some build-up. For a partition $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ of V, the loopless multigraph $G_{\mathcal{P}}$ is defined as follows:

- $G_{\mathcal{P}}$ has k vertices $\{v_1, v_2, \ldots, v_k\}$,
- For every edge of G that is between V_i and V_j, there is an edge in $G_{\mathcal{P}}$ between v_i and v_j.

Our proof of Theorem 3.3.1 will make use of a classical result in graph connectivity, known as the Nash-Williams Theorem, proved independently by Tutte [23] and Nash-Williams [17], which is stated below.

Theorem 3.3.2. *A graph G has k edge-disjoint spanning trees if and only if for every partition $\mathcal{P}$ of $V(G)$, the multigraph $G_{\mathcal{P}}$ has at least $k(|\mathcal{P}| - 1)$ edges.*

This is a very general result and we only need the case $k = 3$, which we state as a corollary.

Corollary 3.3.3. *A graph G has 3 edge-disjoint spanning trees if and only if for every partition $\mathcal{P}$ of $V(G)$, the multigraph $G_{\mathcal{P}}$ has at least $3|\mathcal{P}| - 3$ edges.*

Lemma 3.3.4. *Let G be a connected graph with $|G| > 1$ and let $E_1 \subseteq E(G)$. Then E_1 is connected and spanning if and only if $|G/E_1| = 1$.*

Proof. If E_1 has more than one component, contracting it results more than one vertex.

Also if $v \in V(G) \setminus V(E_1)$ and $u \in V(E_1)$, then u and v are two distinct vertices in G/E_1. On the other hand, if E_1 is connected and spanning, then clearly G/E_1 is a single vertex, possibly with some loops. □

If G is loopless, the graph $2G$ is obtained by adding an edge in parallel to every edge of G, so that every parallel class doubles in size. Motivated by Theorem 3.3.2 and in order to prove Theorem 3.3.1, we define a graph G to be k-*Nash-Williams* if $E(G)$ can be partitioned into $E(G) = E_1 \uplus E_2 \uplus \cdots \uplus E_k$ such that $G/(E \setminus E_i)$ is a single vertex (possibly with some loops) for $i \in \{1, 2, \ldots, k\}$. The following lemma enables us to put Corollary 3.3.3 into use.

Lemma 3.3.5. *Let G be a connected and simple graph. Then G is 3-Nash-Williams if and only if $2G$ has three edge-disjoint spanning trees.*

Proof. Let G be 3-Nash-Williams with $E(G) = E_1 \uplus E_2 \uplus E_3$. Put $T_1 = E_1 \cup E_2$, $T_2 = E_1 \cup E_3$, and $T_3 = E_2 \cup E_3$. Then $|G/T_i| = 1$ for $1 \leqslant i \leqslant 3$. By Lemma 3.3.4, the graphs T_i are connected and spanning, and thus, each contain a spanning tree of G. Furthermore each edge of G is in exactly two of the T_i's, which means that $E(2G)$ is the disjoint union of T_i's, and thus $2G$ has three edge-disjoint spanning trees.

For the other direction, suppose that $2G$ has three edge-disjoint spanning trees T_1, T_2, and T_3. These three trees may not cover all the edges of $2G$. Let $R = E(2G) \setminus (E(T_1) \cup E(T_2) \cup E(T_3))$. For an edge e of $2G$, let $f(e)$ be the unique edge that is parallel to e. Partition R as follows:

$$R_1 = \{e \in R \mid f(e) \in T_1\},$$

$$R_2 = \{e \in R \mid f(e) \in T_2\},$$

$$R_3 = \{e \in R \mid f(e) \in T_3\},$$

$$R_4 = \{e \in R \mid f(e) \in R\}.$$

Edges in R_4 come in parallel pairs, so let $R_4 = R_5 \uplus R_6$, where R_5 and R_6 each contain one edge from each parallel class of R_4. Finally let

$$F_1 = T_1 \cup R_2 \cup R_5,$$

$$F_2 = T_2 \cup R_3 \cup R_6,$$

$$F_3 = T_3 \cup R_1.$$

Note that F_i's may no longer be trees, but each one is connected and spanning, and they all form a partition the edge set of $2G$. Furthermore, parallel edges do not belong to the same F_i. Put

$$E_1 = \{e \in E(2G) \mid e \in F_1 \text{ and } f(e) \in F_2\},$$

$$E_2 = \{e \in E(2G) \mid e \in F_1 \text{ and } f(e) \in F_3\},$$

$$E_3 = \{e \in E(2G) \mid e \in F_2 \text{ and } f(e) \in F_3\}.$$

Then $E(2G) = E_1 \uplus E_2 \uplus E_3$. Also $E_1 \cup E_2 = \{e \in E(2G) \mid e \in F_1 \text{ or } f(e) \in F_1\}$, so $E_1 \cup E_2$ is an isomorphic copy of F_1 in G, and thus is a connected spanning subgraph of G. By Lemma 3.3.4, we conclude that $G/(E_1 \cup E_2)$ is a single vertex. A similar argument applies to the other two contractions, and thus G is 3-Nash-Williams. □

The following proposition is a major step towards the proof of Theorem 3.3.1.

Proposition 3.3.6. *Every simple graph G with at least $\frac{3}{2}(n-1)$ edges has a 3-Nash-Williams subgraph.*

Proof. For $n \leqslant 2$, every simple graph has automatically fewer than $\frac{3}{2}(n-1)$ edges, and for $n = 3$, the only simple graph with 3 edges is K_3 which is itself 3-Nash-Williams. Now let G be a smallest counterexample. This implies that G is connected and G itself is not 3-Nash-Williams. By Lemma 3.3.5, the graph $2G$ does not have 3 edge-disjoint spanning trees. By Theorem 3.3.2, it has a partition $\mathcal{P}$ of size p such that $2G_{\mathcal{P}}$ has fewer than $3(p-1)$ edges, and thus $G_{\mathcal{P}}$ has fewer than $\frac{3}{2}(p-1)$ edges. Note that $p > 1$. Let $X_1, X_2, \ldots, X_p$ be the partition sets, and consider $G[X_i]$ for $1 \leqslant i \leqslant p$.

Lemma 3.3.7. $|E\left(G[X_i]\right)| < \frac{3}{2}(|X_i| - 1)$

Proof. Suppose the lemma fails, so that for some $1 \leqslant i \leqslant p$, the induced graph $G[X_i]$ has at least $\frac{3}{2}(|X_i| - 1)$ edges. Since $G[X_i]$ is smaller than G, it is not a counterexample to Proposition 3.3.6, and thus has some 3-Nash-Williams subgraph N, but N is a subgraph of G as well; a contradiction. □

We can now finish the proof of Proposition 3.3.6. Considering the partition $\mathcal{P}$, there are two types of edges in G:

1. Edges between two different sets in $\mathcal{P}$, and
2. Edges inside a set in $\mathcal{P}$.

By the discussion before Lemma 3.3.7, there are fewer than $\frac{3}{2}(p-1)$ edges of first type, and by Lemma 3.3.7, there are fewer than $\sum_{i=1}^{p} \frac{3}{2}(|X_i| - 1)$ edges of second type. Thus G

has fewer than $\frac{3}{2}(p-1)+\sum_{i=1}^{p}\frac{3}{2}(|X_i|-1)=\frac{3}{2}p-\frac{3}{2}+\frac{3}{2}n-\frac{3}{2}p=\frac{3}{2}(n-1)$ edges, a contradiction.

□

We are now ready to prove the first major result of this chapter, Theorem 3.3.1.

Proof of Theorem 3.3.1. Let G be a smallest $(K_4,3)$-negative graph. First we will establish some basic facts about G, listed as bullet points below.

- **G is loopless.** Clearly loops do not contibute to the creation of a K_4 minor.
- **G has no parallel edges.** Let e and f be two edges in parallel between some vertices u and v, and let $G' = G/e \setminus f \cong G/f \setminus e$. Since G' is smaller than G, it is $(K_4,3)$-positive, say, with $E(G') = E_1' \uplus E_2' \uplus E_3'$. Putting $E_1 = E_1' \cup e$, letting $E_2 = E_2' \cup f$, and setting $E_3 = E_3'$, we see that the extra edges e and f get contracted in $G/(E_1 \cup E_2)$ and get reduced to loops in $G/(E_1 \cup E_3)$ and $G/(E_2 \cup E_3)$. This shows that G is $(K_4,3)$-positive as well, a contradiction.
- **G is connected.** Let $G = G' \uplus G''$ bet he disjoint union of two graphs. Since G' and G'' are smaller than G, they are $(K_4,3)$-positive, say, with $E(G') = E_1' \uplus E_2' \uplus E_3'$ and $E(G'') = E_1'' \uplus E_2'' \uplus E_3''$. Putting $E_1 = E_1' \cup E_1''$, letting $E_2 = E_2' \cup E_2''$, and setting $E_3 = E_3' \cup E_3''$ shows that G is $(K_4,3)$-positive as well, a contradiction.
- **δ(G) is greater than one.** It is not hard to see that pendant vertices do not contribute to the creation of a K_4 minor.

Showing that $\delta(G) = 2$ is impossible is more involved, so we present a more detailed reasoning. For a contradiction, suppose that there are vertices v, x, and y such that $N(v) = \{x,y\}$.

If $xy \in E(G)$, then vxy is a triangle. Let $G' = G/\{vx,vy\}$. Since G' is smaller than

G, it is $(K_4, 3)$-positive, say, with $E(G') = E'_1 \uplus E'_2 \uplus E'_3$. Putting $E_1 = E'_1 \cup \{vx\}$, letting $E_2 = E'_2 \cup \{vy\}$, and setting $E_3 = E'_3 \cup \{xy\}$, shows that G is $(K_4, 3)$-positive as well; a contradiction.

If $xy \notin E(G)$, let $G' = G/vx$. Note that $xy \in E(G')$. Since G' is smaller than G, it is $(K_4, 3)$-positive, say, with $E(G') = E'_1 \uplus E'_2 \uplus E'_3$. Without loss of generality, we may suppose that $xy \in E'_1$. Put $E_1 = E'_1 \cup \{vx, vy\}$, let $E_2 = E'_2$, and set $E_3 = E'_3$. We can easily verify that $G/(E_1 \cup E_2) = G'/(E'_1 \cup E'_2)$ and $G/(E_1 \cup E_3) = G'/(E'_1 \cup E'_3)$. The third graph $G/(E_1 \cup E_2)$ has two edges vx and vy in place of one edge xy of $G'/(E'_1 \cup E'_2)$, but vx and vy are in series, so they will not contribute to the creation of a K_4 minor. This proves that $\delta(G) \geqslant 3$.

We have shown that G is simple and connected and satisfies $\delta(G) \geqslant 3$. By Proposition 3.3.6, the graph G has a 3-Nash-Williams subgraph H with $E(H) = E''_1 \uplus E''_2 \uplus E''_3$ such that $H/(E(H) \smallsetminus E_i)$ is a single vertex. Let $G' = G/H$. Since G' is smaller than G, it is $(K_4, 3)$-positive, say, with $E(G') = E'_1 \uplus E'_2 \uplus E'_3$. Putting $E_1 = E'_1 \cup E''_1$, letting $E_2 = E'_2 \cup E''_2$, and setting $E_3 = E'_3 \cup E''_3$, shows that G is $(K_4, 3)$-positive as well; a contradiction. □

A *cactus* is a connected graph where every block is an edge, two parallel edges, or a cycle. Examining the proof of Theorem 3.3.1 more closely, we can be more specific about the contractions $G/(E_1 \cup E_2)$, $G/(E_1 \cup E_3)$, and $G/(E_2 \cup E_3)$. The following corollary may be proved using a similar argument.

Corollary 3.3.8. *Every graph G is $(K_4, 3)$-positive such that $G/(E_1 \cup E_2)$, $G/(E_1 \cup E_3)$, and $G/(E_2 \cup E_3)$ are cacti with loops.*

3.4. Second Question

Replacing the K_4 in Conjecture 3.2.1 with $K_{2,3}$, we ask the following: Is it true that every graph is $(K_{2,3}, 2)$-positive? The answer to this question is negative, which is the second major result of this chapter.

Theorem 3.4.1. *There exists a graph G such that for every partition $E(G) = E_1 \uplus E_2$ we have $G/E_1 \geqslant_m K_{2,3}$ or $G/E_2 \geqslant_m K_{2,3}$.*

Note that $K_{2,3}$ is incomparable to K_4 in the minor relation, so Theorem 3.4.1 does not resolve Conjecture 3.2.1.

Proof. Consider the graph H given in Figure 3.1. There are three pairwise internally-disjoint (x, y)-paths in H. Let the one containing u' be named P_t, the one containing v' be named P_b, and the one with length three be named P_m. There are two pendant vertices: u and v. We treat H like an edge between u and v. We now take the complete bipartite graph $K_{2,3}$ with u_1 and u_2 on one side and v_1, v_2, and v_3 on the other side, and replace each of its six edges with a copy of H. The H-copy between u_i and v_j is called $H_{i,j}$. We name the resulting graph G.

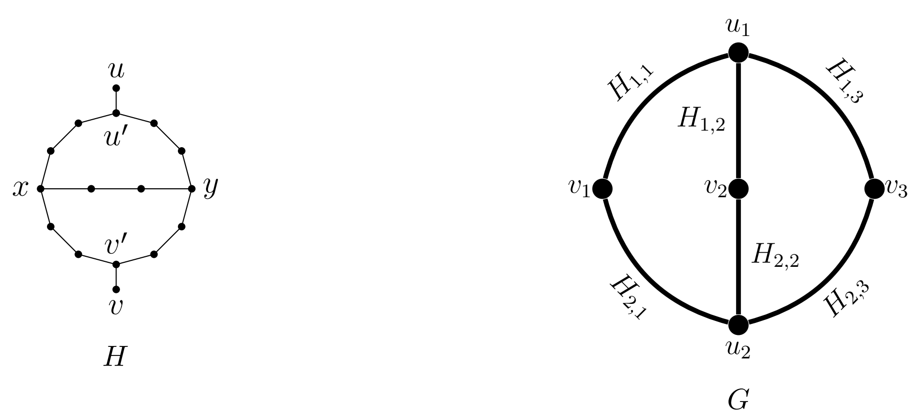
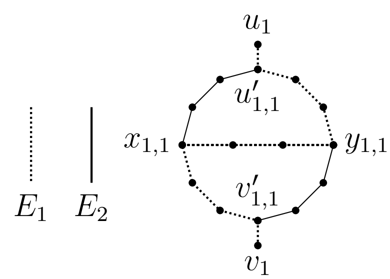


Figure 3.1: The structure of G.

There may be, and in fact there are, many graphs that satisfy the conditions of Theorem 3.4.1. We prove that G is one such graph. Suppose that $E(G) = E_1 \cup E_2$ and consider G/E_1. In at least one copy of H, say $H_{1,1}$, the edges of E_1 must contain an $H_{1,1}$-path from u_1 to v_1, otherwise G/E_1 has a $K_{2,3}$-minor, and the conclusion holds. So without loss of generality, suppose that E_1, restricted to $H_{1,1}$, has a path P from $u'_{1,1}$ to $v'_{1,1}$. There are four cases regarding P:

Case 1: $P_m \subseteq P$.

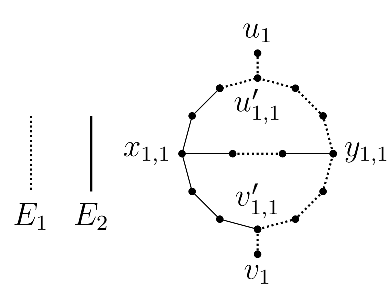


In this case we have $P_m \subseteq P \subseteq E_1$, which implies that:

- $E_2 \cap P_m = \varnothing$,
- $|E_2 \cap P_t| \leqslant 3$, and
- $|E_2 \cap P_b| \leqslant 3$.

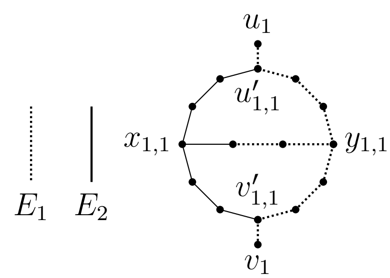
We can see that G/E_2 has three pairwise internally-disjoint $(x_{1,1}, y_{1,1})$-paths, each of length at least two, which implies that $G/E_2 \geqslant_m K_{2,3}$.

Case 2: $P_m \nsubseteq P$, but each of $E_2 \cap P_t$, $E_2 \cap P_m$, and $E_2 \cap P_b$ has cardinality at least 2.



In this case G/E_1 has three internally-disjoint $(x_{1,1}, y_{1,1})$-paths, each of length at least two, which means that $G/E_1 \geqslant_m K_{2,3}$.

Case 3: $P_m \notin P$, and $|E_2 \cap P_m| \leqslant 1$.

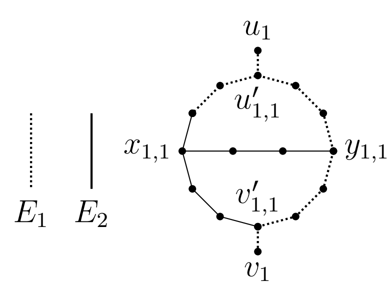


In this case we have:

- $|E_2 \cap P_t| \leqslant 3$,
- $|E_2 \cap P_b| \leqslant 3$, and
- $|E_2 \cap P_m| \leqslant 1$.

Similarly to Case 1, the graph G/E_2 has three pairwise internally-disjoint $(x_{1,1}, y_{1,1})$-paths, each of length at least two, which implies that $G/E_2 \geqslant_m K_{2,3}$.

Case 4: $P_m \notin P$, and $|E_2 \cap P_t| \leqslant 1$ (the case $|E_2 \cap P_b| \leqslant 1$ is argued similarly).



Consider G/E_2. Contract P_m and P_b to a single vertex and name it z. The graph G/E_2 has three internally-disjoint $(u'_{1,1}, z)$-paths: two in $H_{1,1}$ and one using the other copies of H. These paths have length at least two, which implies that $G/E_2 \geqslant_m K_{2,3}$.

It follows that G has a $K_{2,3}$ minor in every case, which concludes the proof. $\square$

Appendix. Copyright Information

arXiv Publishing Agreement

Bibliography

[1] Abd-El-Barr, M. and Gebali, F., Reliability analysis and fault tolerance for hypercube multi-computer networks, *Information Sciences* **276** (2014), 295–318.

[2] Aubert, J. and Schneider, B., Décomposition de la somme Cartésienne d'un cycle et de l'union de deux cycles Hamiltoniens en cycles Hamiltoniens, *Disc. Math.* **38** (1982), 7–16.

[3] Bouya, F., Mahmoodian, E. S., Shokrian, M., and Tefagh, M., A Highly Symmetric Hamilton Decomposition for Hypercubes, *arXiv Preprint* (2020).

[4] Bouya, F. and Oporowski, B., Seymour's second-neighborhood conjecture from a different perspective, *arXiv Preprint* (2019).

[5] Brantner, J., Brockman, G., Kay, B., and Snively, E., Contributions to Seymour's Second Neighborhood Conjecture, *Involve* **2** (2009), 385–393.

[6] Caccetta, L. and Häggkvist, R., On minimal digraphs with given girth, *Proceedings of the Ninth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, FL)* (1978), 181–187.

[7] Chartrand, G., Geller, D., and Hedetniemi, S., Graphs with forbidden subgraphs, *J. Combin. Theory Ser. B* **10** (1971), 12–41.

[8] Das, S. R., Vaidya, N. H., and Patnaik, L. M., Design and implementation of a hypercube multiprocessor, *Microprocessors and Microsystems* **14** (1990), 101–106.

[9] Demaine, E. D., Hajiaghayi, M., and Mohar, B., Approximation algorithms via contraction decomposition, *Combinatorica* **30(5)** (2010), 533–552.

[10] Ding, G., Oporowski, B., Sanders, D. P., and Vertigan, D., Surfaces, tree-width, clique-minors, and partitions, *J. Combin. Theory Ser. B* **79(2)** (2000), 221–246.

[11] Fisher, D., Squaring a tournament: A proof of Dean's Conjecture, *J. Graph Theory* **23** (1996), 43–48.

[12] Gonçalves, D., Edge partition of planar graphs into two outerplanar graphs, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (New York)* (2005), 504–512.

[13] Kaneko, Y. and Locke, S., The minimum degree approach for Paul Seymour's distance 2 conjecture. *Proceedings of the Thirty-Second Southeastern International Conference on Combinatorics, Graph Theory and Computing (Baton Rouge, LA, 2001).* Congr. Numer. **148** (2001), 201–206.

[14] Kedlaya, K. S., Outerplanar partitions of planar graphs, *J. Combin. Theory Ser. B* **67(2)** (1996), 238–248.

[15] Mollard, M. and Ramras, M., Edge decompositions of hypercubes by paths and by cycles, *Preprint* (2013).

[16] Morgan, E. and Oporowski, B., Bounding tree-width via contraction on the projective plane and torus, *The Electronic Journal of Combinatorics* **22** (2015).

[17] Nash-William, C. St. J. A., Edge-disjoint spanning trees of finite graphs, *J. London Math. Soc.* **36** (1961), 445–450.

[18] Okuda, K. and Song, S. W., Revisiting hamiltonian decomposition of the hypercube, *Proceedings of the Thirteenth Symposium on Integrated Circuits and Systems Design (Washington, DC)* (2000), 55–60.

[19] Ostrouchov, G., Parallel computing on a hypercube: an overview of the architecture and some applications, *Nineteenth Symposium on the Interface of Computer Science and Statistics (Philadelphia, PA)* (1987).

[20] Rai, S., Trahan, J. L., and Smailus, T., Processor allocation in hypercube multiprocessors, *IEEE Transactions on Parallel and Distributed Systems* **6** (1995), 606–616.

[21] Ringel, G., Über drei kombinatorische Probleme am n-dimensionalen Würfel und Würfelgitter, *Abh. Math. Sem. Univ. Hamburg* **20** (1955), 10–15.

[22] Song, S. W., Towards a simple construction method for hamiltonian decomposition of the hypercube, *Discrete Math.* **21** (1995), 297–306.

[23] Tutte, W. T., On the problem of decomposing a graph into n connected factors, *J. London Math. Soc.* **36** (1961), 221–230.

Vita

Farid Bouya was born in 1988 in Tehran, Iran. He finished his undergraduate degree in computer science at Sharif University of Technology-Tehran, Iran in 2012 and earned a master of science degree in mathematics from Louisiana State University in May 2014. He is currently a candidate for the degree of Doctor of Philosophy in mathematics.