

**THE UNIVERSITY OF CALGARY**

**Colouring Generalized Kneser Graphs**

**And Homotopy Theory**

by

**Seema Ali**

**A THESIS**

**SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF MASTER OF SCIENCE**

**DEPARTMENT OF MATHEMATICS AND STATISTICS**

**CALGARY, ALBERTA**

**August, 1998**

**© Seema Ali 1998**



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file Votre référence*

*Our file Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-34938-1

## Abstract

Graph colouring is a classic problem in mathematics. In this thesis we are concerned with finding upper and lower bounds of the chromatic number of specific cases of the Generalized Kneser Graph,  $G = G(n, r, q)$ . Using both graph theory and algebraic topology along with a computer we are able to compute some lower bounds for the chromatic number,  $\chi(G)$ , and in many cases to show that these lower bounds coincide with or are very close to (easily obtainable) upper bounds.

## Acknowledgements

I have many people to thank, so one might want to just skip over to the Table of Contents.

First I would like to acknowledge my husband, Jeremy Coolidge for his support, encouragement, and faith in me no matter what. Thanks Jeremy. Thank you Dr. Peter Zvengrowski, for renewing my interests in mathematics. I was just going to obtain a Masters Degree as a prerequisite to get my Ph.D., but you made me see math in a different light. You also taught me to be a little more patient, thorough and careful with proofs. I would also like to thank Arunas Salkauskas, for his patience and generosity with his knowledge about Maple. Arunas also took the time to explain certain aspects about the computer system at the Mathematics Department. Another person that I would like to acknowledge is Arne Storjohann. If it were not for you allowing me to use your ISmith program, I would not be able to find some of the lower bounds for the larger cases. You also created a program that was catered to my situation. Thank you very much for your time, energy and generosity.

I would like to thank Satoshi Tomoda and his wife Fariba Tomoda. Satoshi, I learned a lot from our math discussions/debates. I hope there will be more debates in the future. You have been a good friend to me and my husband, thanks. Of course I would never forget to thank your wife Fariba Tomoda. After moving to Vancouver I came back to Calgary to work on my thesis, and both Fariba and Satoshi took me into their house (without hesitating) where I basically lived for about a month. Fariba lavished me with her excellent cooking, I never had the same meal twice for the entire time I was there. They made me feel comfortable in their home. I must

say that the two of you have spoiled me, I feel like you guys are a part of my family. I would also like to acknowledge my first office-mate, Laura Marik, who was kind, thoughtful and helped me tremendously to adjust to a new big city. Thanks Laura. Thank you Dr. Hadi Kharaghani, Dr. Dave Cowan, Dr. Shelly Wismath, Dr. David Kaminski, Rex Forsyth and Maria Draper for sacrificing your time to answer all of my questions during my B.Sc. You are all excellent instructors, thank you for your help.

Now for my family. I love all of you. Saqib, Karim, Rahim, Mom, Dad and Shahina. And my sisters-in-law: Jennifer Benis and Pamela Coolidge. You are the best sisters-in-law in the world! Thanks for everything.

# Table of Contents

<b>Approval Page</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Table of Contents</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Historical Background . . . . .	1
1.2 Techniques . . . . .	5
1.3 Overviews of the Chapters . . . . .	7
1.4 Conclusions . . . . .	8
<b>2 Homology Theory and Computational Procedure</b>	<b>10</b>
2.1 Introduction . . . . .	10
2.2 Chain Complexes and Homology . . . . .	11
2.3 Simplicial Complexes . . . . .	17
2.4 Homology of Simplicial Complexes . . . . .	23
2.5 Integral Smith Normal Form for Matrices . . . . .	27
2.6 Computer Algorithm for Homology . . . . .	38
<b>3 Graph Colouring</b>	<b>43</b>
3.1 Introduction . . . . .	43
3.2 Graph Theory and Colourings . . . . .	45
3.3 Generalized Kneser graphs . . . . .	49
<b>4 Homotopy and Relations to Graph Colouring</b>	<b>54</b>
4.1 Introduction . . . . .	54
4.2 Homotopy Theory . . . . .	55
4.3 Connectivity and Two Famous Theorems . . . . .	61
4.4 Recent Developments . . . . .	68
<b>5 Applications to Generalized Kneser Graphs</b>	<b>69</b>
5.1 Preliminaries . . . . .	69
5.2 0-Connectivity . . . . .	72
5.3 Simple connectivity of the Kneser neighbourhood complexes . . . . .	73
5.4 Computations of specific examples . . . . .	81

5.4.1	Computation of the homology of $N(5, 2, 0)$ . . . . .	83
5.4.2	Computation of the Relative Homology of $N(5, 2, 0)$ . . . . .	90
5.5	Summary of Results . . . . .	96
5.6	Comparison with Upper Bounds for Chromatic Numbers . . . . .	102
5.7	Checking Procedures . . . . .	105
<b>A</b>	<b>Homology Program</b>	<b>107</b>
<b>B</b>	<b>Maple Program of the Relative Homology</b>	<b>124</b>
B.1	Examples for the procedures of the Relative Homology program . . .	146
B.1.1	binsearch . . . . .	146
B.1.2	repetition . . . . .	146
B.1.3	qsort . . . . .	147
B.1.4	star . . . . .	148
B.1.5	hash . . . . .	148
B.1.6	createflag . . . . .	148
B.1.7	matrixchoose . . . . .	150
B.1.8	write20 . . . . .	150
B.1.9	knCI . . . . .	151
B.1.10	differential . . . . .	152
B.1.11	pn . . . . .	153
B.1.12	kneser . . . . .	153
B.1.13	NOEDGES . . . . .	154
B.1.14	knngbdsimplex . . . . .	154
B.1.15	knhomology . . . . .	154
<b>C</b>	<b>Ismith Program</b>	<b>156</b>
	<b>Bibliography</b>	<b>158</b>
<b>D</b>	<b>Notation</b>	<b>159</b>
	<b>Bibliography</b>	<b>163</b>

## List of Figures

2.1	Topological circles. . . . .	10
2.2	Topological line segments. . . . .	10
2.9	Simplexes . . . . .	40
2.10	Simplicial complexes . . . . .	41
2.11	Skeleton example. . . . .	42
3.1	Königsberg Bridge Problem . . . . .	44
3.3	$G(4, 1, 0)$ and $G(5, 1, 0)$ respectively . . . . .	49
3.4	$G(4, 2, 0)$ . . . . .	50
3.5	$G(4, 2, 1)$ , two perspectives of the same graph . . . . .	50
3.6	$G(5, 2, 0)$ . . . . .	51
4.1	$N(5, 2, 0)$ , see Figure 3.6 for $G(5, 2, 0)$ . . . . .	65
4.2	$N(4, 2, 0)$ (see Figure 3.4 for $G(4, 2, 0)$ ) . . . . .	65
5.1	$N(5, 2, 0)$ . . . . .	87



# Chapter 1

## Introduction

### 1.1 Historical Background

It was either fate or a string of good luck early in Leonhard Euler's life (1707 - 1783) that embarked him upon a remarkable career in mathematics. There were a few times in his life where he could have had a completely different career. First, his father wanted Leonhard at age 17 to enter the ministry, but once Jean Bernoulli informed him of his son's outstanding talent in mathematics he no longer pressured Leonhard to continue with theology. Second, Euler could not find a job in Basel (his birth town) so he left his native Switzerland to take a position in medicine at the St. Petersburg Academy, which had been established a few years earlier by Empress Catherine I. That very day of his arrival she passed away, which threatened the survival of the Academy. Out of desperation, he almost accepted a naval lieutenantcy. About three years later he acquired the Chair of Natural Philosophy. He eventually succeeded Daniel Bernoulli, a good friend, and married Mademoiselle Gsell, a Swiss lady whose father was a painter brought to Russia by Peter the Great. Third, Euler was blind for the last 17 years of his life. Yet this did not slow him down. In fact his work in mathematics flourished. Students would write down everything he would dictate. He could add extremely large numbers in his head, correctly. Even though his eye sight had diminished, his imagination strengthened. Among his copious mathematical works, which fill some twenty-four volumes, is a paper which solves the Königsberg

bridge problem. This is regarded as the first paper in graph theory.

The British mathematician, James Joseph Sylvester (1814 - 1897) was known for his imaginative and sometimes fantastic choice of terminology. In fact, he has been credited with the term 'graph' (which was derived from the chemical graph notation) as it is used today in mathematics (as well as in this dissertation). Sylvester was also very close friends with Arthur Cayley, an even more famous British mathematician who applied graphs to group theory.

In October 1852 Francis Guthrie, a young mathematician, was colouring "... a map showing the counties of England." (cf.[4] p.150-151) from which the celebrated four-colour problem emerged. A map must be coloured so that any two counties (regions) sharing a common boundary do not share the same colour. It occurred to him that just four colours would suffice, for any (planar) map. Hence the four-colour problem. Francis Guthrie could not solve the problem, so he contacted his brother, Fredrick, and discussed it with him. His brother being a physics student decided to present the problem to his mathematics professor who happened to be a prominent English mathematician at the time, Augustus de Morgan. Both de Morgan and Guthrie independently proved that four colours were necessary. De Morgan was able to take it a step further, proving that it was impossible for five counties to all be adjacent to each other. Unfortunately, de Morgan could not solve the four colour problem, so he informed his students about it as well as Sir William Hamilton (the discoverer of quaternions). Even by 1878 the problem still had not been solved when Cayley presented the members of the London Mathematical Society with it. A number of "proofs" were given over the following ninety years which later were shown to be incorrect. In 1976 Kenneth Appel and Wolfgang Haken of

the University of Illinois announced that they had solved the four-colour problem. However, at first their proof struck a nerve with the mathematical community, which led to much criticism. Their proof was extremely dependent on the output of a computer program. There were so many computations and details that it was overwhelming for a human to check the validity of the output. This changed the concept of a mathematical proof, which was that a mathematician would express his proof through a stream of statements, while another mathematician would be able to check the validity of each step instead of enormous numbers of calculations. Although with time the resistance to their proof has lowered, it was still clouded with some doubt due to "... periodic rumours that a subtle error had been found in the computer program which would render the proof useless." (cf.[4] p.149). Currently it is still unknown if a "standard" proof is obtainable.

Seven years after Guthrie found the four colour problem, Hamilton invented a game known as 'The Traveller's Dodecahedron', 'A Voyage Round the World', or 'The Icosian Game'. The game consisted of twenty vertices/pegs which represented "important places" such as Canton, Brussels, Delhi, etc., on a dodecahedron. The game also required thread which was used to loop around the pegs. The goal of the game was to loop each peg only once. If this was achieved it was called a 'voyage round the world'. Unfortunately, even though Hamilton sold the idea of the game to a dealer in games and puzzles for £25, it did not become very successful. The idea behind this game, however, is now famous in graph theory under the name Hamiltonian circuit.

The Danish mathematician Julius Petersen (1839 - 1910) was the first to discuss (generally) the problem of factoring graphs. His idea of graphs was two dimensional.

He considered them as figures formed by points and lines in a plane. This caused many edges to cross (where there were no points/vertices). He actually discovered a graph that could not be divided into three-disjoint one-factors, contained no leaves and was trivalent, nowadays known as the Petersen graph (cf.[14], p.10, Example 1.1.9). We now leave graph theory and turn to algebraic topology.

Through a series of papers from 1895 to 1904, the great French mathematician Henri Poincaré laid the foundation for algebraic topology. He had

... developed a method for construction of geometrical objects, which he called ‘complexes’ (following Listing), from basic building blocks called ‘cells’. In order to describe how the cells fit together, he adapted Kirchhoff’s technique, replacing the system of linear equations by a matrix. The simplest kinds of cells are the 0-cells (vertices) and the 1-cells(edges); a complex constructed from such cells is a graph, and the matrix Poincaré used to describe how the cells are fitted together in this case is now called the ‘incidence matrix’ of the graph (cf.[2] p.135).

This technique (or theory) became instantly popular. It was included in “... the third volume of the famous *Encyklopädie der Mathematischen Wissenschaften*, a monumental work which was intended to survey all the mathematical knowledge available at the time.” (cf.[2] p.135).

Poincaré was not alone in propagating the spread of topology. The Americans George David Birkhoff and Oswald Veblen also helped spread topology through the United States. Thus there was enormous development in both homology and homotopy theory in the 1900’s.

Since algebraic topology mainly deals with questions in spaces of arbitrary dimension, compared to the two-dimensional case of graph theory, the two subjects tended to grow in quite separate directions in the twentieth century. However, in 1978 the Hungarian mathematician L. Lovász [7] was able to prove a famous conjecture in combinatorial set theory known as the Kneser Conjecture, using tools from algebraic topology. This work leads to a fascinating connection between the two seemingly disparate branches of mathematics, graph colouring and homotopy theory. Generalizations of this can be found in further work of Lovász [1] and stronger versions of some of these theorems in the recent work of Milgram and Zvengrowski [8]. It also is very remarkable that purely combinatorial conjectures such as the Kneser Conjecture (or a generalization called the Erdős Conjecture) can only be proved by homotopy theoretic means, at least to date.

## 1.2 Techniques

There are five main tools used in this dissertation: forming the neighbourhood complex of a graph, homology theory, the Hurewicz Theorem, the Lovász Theorem alluded to above, and computer implementation of homology calculations. The precise mathematical formulations of each of these methods appears in the appropriate chapter, e.g. for the definition of a neighbourhood complex see Chapter 4, Section 4.3. Here we give only a thumbnail sketch of each concept.

A simplicial complex is the  $n$ -dimensional generalization of a “triangular” polyhedron, made up of vertices, edges, triangles, tetrahedra, etc. Given any finite graph, its neighbourhood complex is a certain simplicial complex formed by considering

neighbouring vertices in the graph. Once one has any finite simplicial complex  $X$ , the algebraic tool of homology theory can be applied to analyze it. It gives a finite sequence of (finitely generated) abelian groups  $H_0(X), H_1(X), \dots, H_n(X)$ . This is a straightforward but possibly lengthy computation, and can be implemented on a computer using a fairly standard routine called the integral Smith normal form for a matrix (with integer entries). After we have found the homology of the neighbourhood complex, we can determine a lower bound on the number of colours required to “colour”  $G$  using the Hurewicz Theorem, together with some theorems proved in Chapter 5, and the Lovász Theorem. We shall apply all these tools to the generalized Kneser graphs, a family of graphs which include the classical Kneser graphs as a special case.

The main goal of this thesis is to generate a few generalized Kneser graphs and study their connectivity. We need to take a few smaller steps in order to accomplish this. First, we generate the generalized Kneser graph, called  $G(n, r, q)$ . Second, using Lovász’s construction of the neighbourhood complex, mentioned earlier, we form the neighbourhood complex. Third, the homology algorithm is implemented on a computer to explicitly determine the homology groups of the neighbourhood complex. Fourth, using the Hurewicz Theorem, together with certain theorems proved in Chapter 5, information about the homotopy of the neighbourhood complex is found (in particular its connectivity). Then finally, through the Lovász Theorem, we can obtain applications to colourings and other questions. This is illustrated in the following diagram.

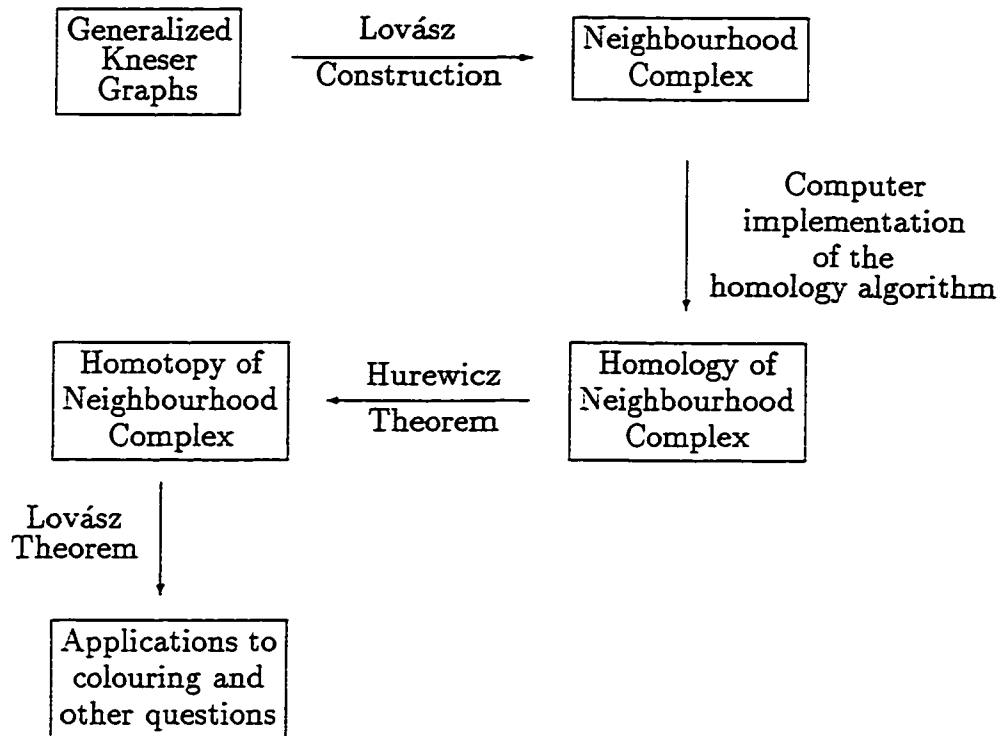


Diagram 1.1: Schematic Flow Chart

### 1.3 Overviews of the Chapters

In Chapter 2, the reader will be given a brief description of homology theory starting with simplicial complexes. Then chain complexes and homology groups as well as the integral Smith normal form of a matrix are defined and described. In Chapter 3, there will be a brief introduction to graph theory, dealing with chromatic numbers and what it means to colour a graph. The generalized Kneser graphs are introduced in this chapter. Homotopy theory and the fundamental groups will appear in Chapter 4, as well as the previously mentioned theorems by Hurewicz and Lovász. In Chapter 5, the generalized Kneser graphs are studied in more detail. Several

theorems giving conditions for their neighbourhood complexes to be connected and simply connected are proved. These theorems are original results, and they enable one to apply the Hurewicz Theorem to then obtain information about the homotopy of these neighbourhood complexes. In particular the connectivity is determined, and the Lovász Theorem is then applied to finding lower bounds for the chromatic number of the generalized Kneser graphs, at least in the cases that could be handled by a computer.

## 1.4 Conclusions

This dissertation contains a good deal of expository material leading to the problem of colouring generalized Kneser graphs  $G(n, r, q)$ . For  $q = 0$  this problem had been completely solved by Lovász [7]. Here we make a beginning on the cases  $q > 0$ . For a sample of the results the author has obtained see Section 5.5 and Section 5.6.

There were certain original theorems that Dr. Peter Zvengrowski and the author worked on together which appear in this dissertation. These would be the following.

1. The Duality Theorem, Theorem 3.3.3 on page 51.
2. Lemma 5.1.6 on page 71.
3. Theorem 5.2.1 on page 72.
4. Lemma 5.3.3 on page 73.
5. Corollary 5.3.4 on page 76.
6. Theorem 5.3.6 on page 77.



7. Theorem 5.3.7 on page 79.

8. Section 5.5 on page 96.

The author wrote the Maple programs deriving her own functions, except for the procedure and function called `binsearch` and `qsort` which are standard procedures in programming. The `ISmith` program was due solely to Arne Storjohann. The author of this dissertation also derived the computational examples generated by the program, which are found throughout the thesis.

## Chapter 2

# Homology Theory and Computational Procedure

### 2.1 Introduction

Topology is sometimes referred to as “rubber geometry” or “rubber sheet geometry”. For example, a square, circle, triangle, or even the outline of a musical note (Figure 2.1) are considered “equivalent” (homeomorphic to be more precise).

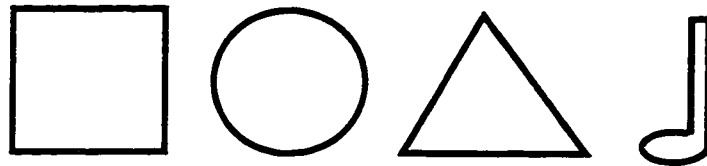


Figure 2.1: Topological circles.

The following four shapes are also homeomorphic to each other.



Figure 2.2: Topological line segments

However, none of the shapes in Figure 2.1 are homeomorphic to any shape in

Figure 2.2. Although statements of this type are intuitively obvious, mathematical proofs may not be easy. A tool that enables one to find such proofs is homology theory. Homology theory creates homology groups. Each space  $X$  has associated to it a sequence of abelian groups  $H_n(X)$ , where  $n \in \mathbb{Z}$  and  $n \geq 0$ , called the homology groups of  $X$ . These are precisely defined in the following sections. Intuitively, however, they measure “ $n$ -dimensional holes” in the space. For example, the shapes in Figure 2.1 all have a single 1-dimensional hole, and satisfy  $H_1(X) \approx \mathbb{Z}$ , while the shapes in Figure 2.2 have no such 1-dimensional hole, and for these  $H_1(X) = 0$ . Thus one obtains a precise proof that the two types are not homeomorphic.

In Section 2.2 the algebraic background for homology is described. In Section 2.3 the fundamental class of spaces to which we shall apply homology theory, the simplicial complexes, is introduced. The previous two sections are combined in Section 2.4 to define the homology of simplicial complexes. Further, linear algebra is applied in Section 2.5, namely the integral Smith normal form, and computer procedures to implement this are given in Section 2.6.

## 2.2 Chain Complexes and Homology

**Definition 2.2.1** A chain complex  $(S_*, \partial)$  is a sequence of abelian groups  $S_n$  and homomorphisms  $\partial_n$  for  $n \in \mathbb{Z}$

$$\cdots \longrightarrow S_{n+1} \xrightarrow{\partial_{n+1}} S_n \xrightarrow{\partial_n} S_{n-1} \longrightarrow \cdots, \quad n \in \mathbb{Z}$$

such that  $\partial_n \partial_{n+1} = 0$  for all  $n \in \mathbb{Z}$ . The homomorphism  $\partial_n$  is called the differential map of degree  $n$ . One calls  $S_n$  the  $n$ -chains.

We can generalize this definition to a chain complex of  $R$ -modules where  $R$  is a commutative ring with unit. However, for this dissertation we will restrict our attention to abelian groups, as in the above definition.

**Remark 2.2.2** Notice that  $\partial_n \partial_{n+1} = 0 \Leftrightarrow \text{im } \partial_{n+1} \subseteq \ker \partial_n$ .

If, for some  $p < q$ ,  $S_n = 0$  for all  $n > q$  or  $n < p$ , then we write:

$$0 \longrightarrow S_q \xrightarrow{\partial_q} S_{q-1} \longrightarrow \cdots \longrightarrow S_{p+1} \xrightarrow{\partial_{p+1}} S_p \longrightarrow 0$$

instead of

$$\cdots \longrightarrow 0 \longrightarrow S_q \xrightarrow{\partial_q} S_{q-1} \longrightarrow \cdots \longrightarrow S_{p+1} \xrightarrow{\partial_{p+1}} S_p \longrightarrow 0 \longrightarrow \cdots$$

**Definition 2.2.3** Let  $(S_*, \partial)$  be a chain complex. We say that  $(S'_*, \partial')$  is a sub-chain complex of  $(S_*, \partial)$  if  $S'_n \subseteq S_n$  and  $\partial'_n = \partial_n|_{S'_n}$  for all  $n \in \mathbb{Z}$ . In this case one can also form the quotient chain complex  $(T_*, \partial'')$  where  $T_n = S_n/S'_n$ , with differential the obvious homomorphism  $S_n/S'_n \rightarrow S_{n-1}/S'_{n-1}$  induced by  $\partial_n$ .

We now give a few examples of chain complexes.

#### Examples 2.2.4

1. Let  $S$  be an abelian group and  $0$  be its additive identity. Consider the sequence

$$\cdots \longrightarrow S \xrightarrow{k} S \xrightarrow{k} S \longrightarrow \cdots$$

where  $k(s) = 0$  for all  $s \in S$ .

Clearly,  $\text{im } k = 0$  and  $\ker k = S$ . Therefore,  $kk(s) = 0$  for all  $s \in S$ . Thus  $kk = 0$ .

Therefore this sequence is a chain complex.

2. Consider  $0 \longrightarrow \mathbb{Z} \xrightarrow{t_m} \mathbb{Z} \xrightarrow{\pi} \mathbb{Z}_m \longrightarrow 0$ , where  $t_m(n) = m \cdot n$  and  $\pi$  is the natural projection with  $\pi(1) = 1$ .

Since  $\pi t_m(n) = \pi(mn) = m \cdot \pi(n) = 0 \in \mathbb{Z}_m$ , we have  $\pi t_m = 0$ .

3. Consider the sequence  $0 \longrightarrow A \xrightarrow{f} B \xrightarrow{g} C \longrightarrow 0$  where  $A, B, C$  are free abelian groups with the bases  $\{a_1, a_2, a_3\}$ ,  $\{b_1, b_2, b_3, b_4\}$  and  $\{c_1, c_2\}$  respectively. Let  $f$  and  $g$  be defined as follows:

$$\begin{aligned} f(a_1) &= 6b_1 + 6b_2 - 12b_3 - 18b_4 \\ f(a_2) &= 9b_2 - 18b_3 - 27b_4 \\ f(a_3) &= -45b_2 - 36b_3 - 27b_4 \end{aligned}$$

$$\begin{aligned} g(b_1) &= -6c_1 \\ g(b_2) &= 3c_1 \\ g(b_3) &= 9c_1 \\ g(b_4) &= -7c_1. \end{aligned}$$

We now show that  $gf = 0$  using the fact that  $f$  and  $g$  are homomorphisms.

$$\begin{aligned} gf(a_1) &= g(6b_1 + 6b_2 - 12b_3 - 18b_4) \\ &= 6g(b_1) + 6g(b_2) - 12g(b_3) - 18g(b_4) \\ &= -36c_1 + 18c_1 - 108c_1 + 126c_1 \\ &= 0, \end{aligned}$$

$$\begin{aligned} gf(a_2) &= g(9b_2 + 18b_3 + 27b_4) \\ &= 9g(b_2) + 18g(b_3) + 27g(b_4) \\ &= 27c_1 + 162c_1 - 189c_1 \\ &= 0, \end{aligned}$$

$$\begin{aligned}
gf(a_3) &= g(-45b_2 + 36b_3 + 27b_4) \\
&= -45g(b_2) + 36g(b_3) + 27g(b_4) \\
&= -135c_1 + 324c_1 - 189c_1 \\
&= 0.
\end{aligned}$$

It follows that  $gf(n_1a_1 + n_2a_2 + n_3a_3) = n_1 \cdot gf(a_1) + n_2 \cdot gf(a_2) + n_3 \cdot gf(a_3) = 0$  where  $n_1, n_2, n_3 \in \mathbb{Z}$ . Therefore,  $gf = 0$  and we have a chain complex.

This can be done in a more condensed manner, by taking advantage of matrices.

The matrices associated to  $f$  and  $g$  respectively are:

$$M_f = \begin{bmatrix} 6 & 6 & -12 & -18 \\ 0 & 9 & -18 & -27 \\ 0 & -45 & -36 & -27 \end{bmatrix}, M_g = \begin{bmatrix} -6 & 0 \\ 3 & 0 \\ 9 & 0 \\ -7 & 0 \end{bmatrix}.$$

One can easily check that  $M_f \cdot M_g = 0_{3 \times 2}$ , where  $0_{m \times n}$  is the  $m \times n$  zero matrix, and since  $M_f \cdot M_g = M_{gf}$ , it follows that  $gf = 0$ .

**Definition 2.2.5** A sequence of two homomorphisms (of groups)  $A \xrightarrow{f} B \xrightarrow{g} C$  is exact at  $B$  if  $\text{im } f = \ker g$ . A sequence of abelian groups  $\{S_n\}_{n \in \mathbb{Z}}$  and homomorphisms  $\{\partial_n\}_{n \in \mathbb{Z}}$ ,

$$\cdots \longrightarrow S_{n+1} \xrightarrow{\partial_{n+1}} S_n \xrightarrow{\partial_n} S_{n-1} \longrightarrow \cdots, \quad n \in \mathbb{Z}$$

is exact if it is exact at each  $S_n$ . That is,  $\text{im } \partial_{n+1} = \ker \partial_n$  for all  $n \in \mathbb{Z}$ .

A short exact sequence is an exact sequence of the form

$$0 \rightarrow A \xrightarrow{f} B \xrightarrow{g} C \rightarrow 0.$$

Notice that  $f$  is then necessarily monic (1-1) and  $g$  is necessarily epic (onto).

### Examples 2.2.6

1.  $0 \rightarrow A \rightarrow 0$  is exact if and only if  $A = 0$ .

Proof: The sequence  $0 \xrightarrow{f} A \xrightarrow{g} 0$  is exact.

$$\Leftrightarrow \operatorname{im} f = \ker g, \quad (\text{definition of exactness})$$

$$\Leftrightarrow \ker g = 0, \text{ since } \operatorname{im} f = 0,$$

$$\Leftrightarrow A = 0, \text{ since } \ker g = A.$$

□

2.  $0 \rightarrow A \xrightarrow{f} B \rightarrow 0$  is exact if and only if  $f$  is an isomorphism.

Proof: The sequence  $0 \xrightarrow{g} A \xrightarrow{f} B \xrightarrow{h} 0$  is exact

$$\Leftrightarrow \operatorname{im} g = \ker f \text{ and } \operatorname{im} f = \ker h,$$

$$\Leftrightarrow 0 = \ker f \text{ and } \operatorname{im} f = B, \text{ since } \operatorname{im} g = 0, \ker h = B,$$

$$\Leftrightarrow f \text{ is monic and epic,}$$

$$\Leftrightarrow f \text{ is an isomorphism.}$$

□

3. For any homomorphism  $f : A \rightarrow B$ , there is a short exact sequence

$$0 \rightarrow \ker f \xrightarrow{i} A \xrightarrow{f'} \operatorname{im} f \rightarrow 0$$

Here  $i$  is the inclusion ( $ix = x$  for all  $x \in \ker f$ ) and  $f'$  is the corestriction of  $f$  to its codomain, i.e.  $f'(a) = f(a)$  for all  $a \in A$ .

Proof: Assume  $f : A \rightarrow B$  is a homomorphism and consider the sequence  $0 \rightarrow \ker f \xrightarrow{i} A \xrightarrow{f'} \operatorname{im} f \rightarrow 0$ , with  $i$  and  $f'$  as previously described. Clearly  $i$  is monic since  $i$  is the inclusion map. Similarly  $f'$  is epic, since  $\operatorname{im} f' = \operatorname{im} f$ . It follows that the sequence is exact at  $\ker f$  and at  $\operatorname{im} f$ .

All that remains is to show that the sequence is exact at  $A$ . Now  $\operatorname{im} i = \ker f$  and  $\ker f' = \ker f$ , since  $i$  is the inclusion map and  $f'(a) = f(a)$  for all  $a \in A$ . Thus  $\operatorname{im} i = \ker f'$ , and the sequence is therefore exact at  $A$ .  $\square$

4. If  $A$  is a subgroup of  $B$  then

$$0 \rightarrow A \xrightarrow{i} B \xrightarrow{\pi} B/A \rightarrow 0$$

is exact, where  $i$  = inclusion and  $\pi$  = natural projection.

Proof: Let  $i$  be an inclusion map and  $\pi$  be the natural projection where  $0 \rightarrow A \xrightarrow{i} B \xrightarrow{\pi} B/A \rightarrow 0$ .

By their definitions,  $i$  is monic and  $\pi$  is epic. Since  $\ker \pi = A = \operatorname{im} i$ , the sequence is short exact.  $\square$

**Definition 2.2.7** If  $(S_*, \partial)$  is a chain complex, then  $\ker \partial_n$  is called the group of **n-cycles** and denoted  $Z_n(S_*, \partial)$ ;  $\operatorname{im} \partial_{n+1}$  is called the group of **n-boundaries** and is denoted by  $B_n(S_*, \partial)$ . The **nth homology group** of this complex is  $H_n(S_*, \partial) = Z_n(S_*, \partial)/B_n(S_*, \partial)$ .

One often writes just  $Z_n(S_*)$  or simply  $Z_n$  for  $Z_n(S_*, \partial)$ , and similarly  $B_n(S_*)$  or  $B_n$  for  $B_n(S_*, \partial)$ , and  $H_n(S_*)$  or  $H_n$  for  $H_n(S_*, \partial)$  when it is clear that  $(S_*, \partial)$  is the only chain complex under consideration.



If  $z \in Z_n$ , then  $z + B_n \in H_n$  is called the **homology class** of  $z$  and it is denoted by  $[z]$ .

**Remark 2.2.8** Notice that for this definition to make sense, one needs  $B_n(S_*, \partial) \subseteq Z_n(S_*, \partial)$ , and we have already seen that this is so (cf. Remark 2.2.2).

**Theorem 2.2.9** A chain complex  $(S_*, \partial)$  is exact if and only if  $H_n(S_*, \partial) = 0$  for all  $n \in \mathbb{Z}$ .

Proof:  $Z_n = B_n$  if and only if  $\ker \partial_n = \text{im } \partial_{n+1}$ . □

Intuitively, this means that homology groups “measure” deviation of a chain complex from being exact. It is also convenient, recalling Definition 2.2.3 to define the “relative homology” of a chain complex  $(S_*, \partial)$ , with a given sub-chain complex  $(S'_*, \partial')$ .

**Definition 2.2.10** Let  $(S'_*, \partial')$  be a sub-chain complex of  $(S_*, \partial)$ . The relative homology of the pair  $(S_*, S'_*)$  is defined as  $\mathbf{H}_n(\mathbf{S}_*, \mathbf{S}'_*) = H_n(S_*/S'_*)$ .

**Example 2.2.11** If  $L = M$  then  $H_n(L, M) = 0$  for all  $n \in \mathbb{Z}$ .

For a less trivial example see Example 2.5.8.

## 2.3 Simplicial Complexes

Intuitively, a simplex is an  $n$ -dimensional generalization of a point (0-simplex), a line segment (1-simplex), a solid triangle (2-simplex), a solid tetrahedron (3-simplex), etc. Any subset of the vertices of a simplex  $\sigma$  determines a smaller simplex (or  $\sigma$  itself) called a face of  $\sigma$ . One thinks of the empty set as a  $(-1)$ -simplex. The following figure illustrates simplexes of dimensions 0,1,2,3 respectively.

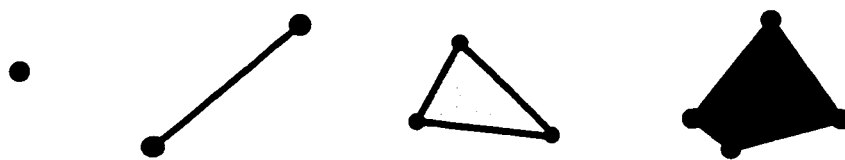


Figure 2.3

Notice that in order to illustrate a solid three dimensional shape we used a darker shading compared to a solid two dimensional shape.

A simplicial complex is a space made up of simplexes as its building blocks, which fit together “nicely”, meaning that the intersection of any two simplexes is a common face of each (or is empty). Its dimension is the maximal dimension of any of its simplexes.

**Example 2.3.1** The shapes shown here (Figure 2.4) are examples of a 1-dimensional, a 2-dimensional, and a 3-dimensional simplicial complex respectively.

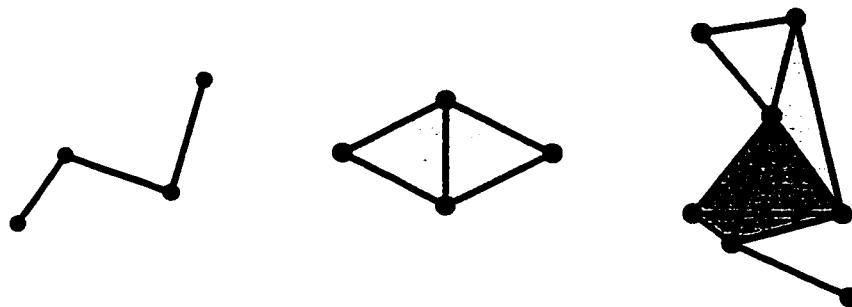


Figure 2.4

**Example 2.3.2** None of the three shapes shown in Figure 2.5 on page 19 is a simplicial complex.

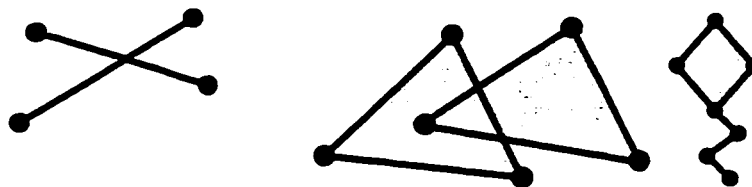


Figure 2.5

Further subdivisions could be made to transform them into simplicial complexes (see Figure 2.6). Also keep in mind that these subdivisions are not unique.

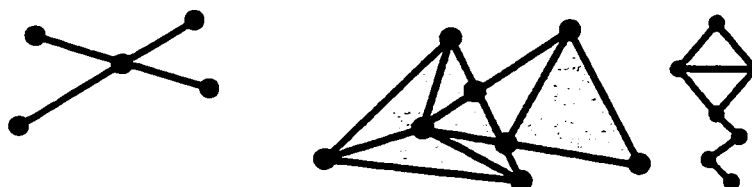


Figure 2.6

Notice that any simplicial complex is determined by

1. a set of vertices,
2. the specification of those subsets of vertices that form the simplexes.

This leads to the following definition:

**Definition 2.3.3** For any set  $V$ , let  $\mathbf{P}(V)$  denote the power set of  $V$ , which is the set of all subsets of  $V$  (including the empty set). An abstract simplicial complex  $K$  is a pair  $K = (V, \Sigma)$ , where  $\Sigma \subseteq \mathbf{P}(V)$ , such that

1. For all  $v \in V$ ,  $\{v\} \in \Sigma$ .

2. If  $\tau \in \Sigma$  and  $\sigma \subseteq \tau$ , then  $\sigma \in \Sigma$ .

The elements of  $V$  are called the vertices of  $K$ . The elements of  $\Sigma$  are called the simplexes of  $K$ . Note that an abstract simplicial complex is finite if  $|V| < \infty$ . A simplex with  $r + 1$  vertices is called an  $r$ -simplex. A 1-simplex is also known as an edge.

**Definition 2.3.4** A simplex  $\{v_0, \dots, v_n\}$  is a maximal simplex of a simplicial complex  $K = (V, \Sigma)$  if  $\{v_0, \dots, v_n, w\}$  is not a simplex in  $K$  for any  $w \in V$  where  $w \neq v_i$  for all  $0 \leq i \leq n$ .

**Remark 2.3.5** It is often convenient to describe an abstract simplicial complex by giving only the maximal simplexes; then all simplexes are obtained from these and their subsets (faces).

**Definition 2.3.6** A subcomplex  $K' = (V', \Sigma')$  of a simplicial complex  $K = (V, \Sigma)$  is a simplicial complex such that  $V' \subseteq V$  and  $\Sigma' \subseteq \Sigma$ .

Next we define the geometric idea behind the abstract simplicial complex.

**Definition 2.3.7** An ordered set of points  $\{v_0, v_1, \dots, v_n\} \subset \mathbb{R}^n$  is **affine independent** if  $\{v_1 - v_0, v_2 - v_0, \dots, v_n - v_0\}$  is a linearly independent subset of the real vector space  $\mathbb{R}^n$ .

**Remark 2.3.8** It is not hard to prove that this definition is independent of the ordering.

**Definition 2.3.9** Let  $\{v_0, \dots, v_q\}$  be an affine independent set of some euclidean space. A  $q$ -simplex  $s = (v_0, \dots, v_q)$  is the smallest convex subset containing the vertices  $v_0, \dots, v_q$ , also called the convex hull of  $\{v_0, \dots, v_q\}$ . (See Figures 2.3, page 18 and 2.9, page 40 for some basic examples.) For any subset  $\{w_0, \dots, w_r\} \subseteq \{v_0, \dots, v_q\}$ , the convex hull of  $\{w_0, \dots, w_r\}$  is called a face of  $s$ . It is convenient to consider the empty set,  $\emptyset$ , as a  $(-1)$ -simplex.

**Definition 2.3.10** A finite geometric simplicial complex a finite collection of simplexes in  $\mathbb{R}^N$  such that for any simplexes  $\sigma, \tau \in X$ ,  $\sigma \cap \tau$  is a common face of each.

**Definition 2.3.11** Let  $\mathcal{K}$  denote the set of all finite geometric simplicial complexes,  $\mathcal{K}^a$  denote the set of all finite abstract simplicial complexes. Any geometric simplicial complex  $X \in \mathcal{K}$  determines an underlying abstract simplicial complex  $K = U(X)$  where  $U : \mathcal{K} \rightarrow \mathcal{K}^a$ . The vertices  $V$  of  $U(X) = (V, \Sigma)$  are the same as the vertices of  $X$ , and a collection  $\{v_0, \dots, v_n\}$  of vertices in  $V$  forms a simplex of  $\Sigma$  if and only if there is a geometric simplex  $\sigma$  of  $X$  with vertices  $v_0, \dots, v_n$ .

**Definition 2.3.12** Given an abstract simplicial complex  $K$ , we say that a simplicial complex  $X$  is a geometric realization of  $K$  if  $U(X) = K$ .

While it may be very difficult to visualize a geometric simplicial complex, especially in dimensions greater than three, the underlying abstract simplicial complex gives a simple combinatorial model. In this connection, it is interesting to note the following theorem.

**Theorem 2.3.13** *Any finite abstract simplicial complex  $K$  admits a geometric realization  $X$  in  $\mathbb{R}^N$  for some  $N \in \mathbb{N}$ , i.e.  $U(X) = K$ .*

A proof of Theorem 2.3.13 can be found in [11], p.142. One writes  $|K|$  (or simply  $X$ ) for the topological space determined by  $K$ , as a subspace of  $\mathbb{R}^n$ . This space is unique up to homeomorphism.

**Definition 2.3.14** *Let  $K = (V, \Sigma)$  be any abstract simplicial complex. The  $r$ -skeleton of  $K$  is  $K^{(r)} := (V, \Sigma^{(r)})$  where  $\Sigma^{(r)}$  contains all  $j$ -simplexes for  $j \leq r$ . It is also a simplicial complex (subcomplex of  $K$ ). A similar definition applies to geometric simplicial complexes.*

Notice that the 1-skeleton can also be regarded as a graph, since it only contains vertices and edges. In order to construct a hollow triangle, i.e. a (topological) circle, we need three 1-simplexes arranged appropriately. For details on this example and for further examples see Figure 2.10 on page 41.



Illustrations have been frequently applied in mathematics to help “visualize” a mathematical situation. An old saying goes: “a picture is worth a thousand words,” and it rings true in mathematics as well. Suitable diagrams are often helpful in visualizing the large amount of algebra that is associated with these geometric objects.

**Example 2.3.15** See Figure 2.11, on page 42, for the  $r$ -skeleta of the simplicial complex  $K$ , where  $K$  is indicated by the next diagram.

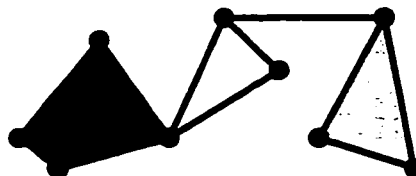


Figure 2.7

## 2.4 Homology of Simplicial Complexes

It is often quite difficult to visualize, let alone prove, whether or not two given simplicial complexes are homeomorphic. This might occur if the simplicial complexes are too complicated to picture, in a dimension higher than three, or the characteristics of the complexes in question have never been researched, and so on. Whatever the reason is, homology groups will furnish algebraic invariants and thereby replace intuition with precise mathematical proofs. Homology groups can be effectively computed, as we shall see in Section 2.5. First, however, in order to determine the homology of simplicial complexes, we need to set up some definitions.

**Definition 2.4.1** Let  $(S_*, \partial)$  be a chain complex. We say, for any two cycles  $x, y \in Z_n(S_*, \partial)$ , that  $x$  is **homologous to  $y$**  ( $x \sim y$ ) if and only if  $y - x \in B_n(S_*, \partial)$ . We also write  $x + B_n(S_*, \partial) = [x]$  for the equivalence class of all cycles homologous to  $x$ .

**Definition 2.4.2** Let  $K = (V, \Sigma)$  be an abstract simplicial complex and suppose the vertices  $V$  are given some simple order  $<$ . For each  $n$ -simplex  $\sigma \in \Sigma$ , one can

uniquely write  $\sigma = \{v_0, \dots, v_n\}$  with  $v_0 < v_1 < \dots < v_n$ . We then define the  $n$ -chains of  $K$ ,  $C_n(\mathbf{K})$ , to be the free abelian group having as generators all symbols  $\langle v_0, \dots, v_n \rangle$ , one for each  $\sigma \in \Sigma$ , where  $\sigma = \{v_0, \dots, v_n\}$  and  $v_0 < v_1 < \dots < v_n$ . Writing  $\langle \sigma \rangle = \langle v_0, \dots, v_n \rangle$ , we thus have

$$C_n(\mathbf{K}) = \left\{ \sum_{i=1}^m a_i \langle \sigma_i \rangle : a_i \in \mathbb{Z}, m \geq 1, |\sigma_i| = n+1, \sigma_i \in \Sigma, 1 \leq i \leq m \right\}.$$

**Definition 2.4.3** Let  $K$  be an abstract simplicial complex with some simple order given on its vertices  $V$ , as in Definition 2.4.2. We define the map  $d_n : C_n(K) \rightarrow C_{n-1}(K)$  on the generators by

$$d_n \langle v_0, \dots, v_n \rangle = \sum_{i=0}^n (-1)^i \langle v_0, \dots, \hat{v}_i, \dots, v_n \rangle,$$

where  $\hat{v}_i$  represents the deleted vertex  $v_i$ , and extend this linearly to all of  $C_n(K)$ .

It is a standard fact that  $d_{n-1}d_n = 0$  (cf.[11], p.44, Theorem 7.11), so  $C_*(K)$  is a chain complex (cf.[11], p.65, Theorem 4.6).

**Definition 2.4.4** The homology of a simplicial complex  $K$  is defined to be the homology of  $C_*(K)$ , i.e.  $H_n(K) := H_n(C_*(K))$ .

It is a deep theorem of algebraic topology, the so-called ‘‘Invariance Theorem,’’ that the homology groups  $H_*(K)$  depend only on  $X = |K|$ , which means that the particular decomposition of  $X$  into a simplicial complex and the order chosen for the vertices have no effect on  $H_*(K)$ . One therefore is justified in writing  $H_*(X) = H_*(K)$ .

**Example 2.4.5** We calculate the homology of  $S^1$ , a circle. Let  $X$  be the geometric simplicial complex shown in the next figure.



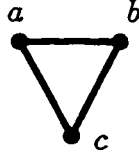


Figure 2.8

Let  $K$  be the abstract simplicial complex of  $X$ , with the vertex order  $a < b < c$ . Therefore,  $K = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}\}$ , and consider

$$C_*(K) : \cdots \longrightarrow C_3(K) \xrightarrow{d_3} C_2(K) \xrightarrow{d_2} C_1(K) \xrightarrow{d_1} C_0(K) \xrightarrow{d_0} 0. \quad (2.1)$$

Let us determine a basis for the  $n$ -chains of  $K$ ,  $C_n(K)$  where  $n \geq 0$ . A basis for  $C_0(K)$  is  $\{\langle a \rangle, \langle b \rangle, \langle c \rangle\}$ . A basis for  $C_1(K)$  is  $\{\langle a, b \rangle, \langle b, c \rangle, \langle a, c \rangle\}$ , and we also have  $C_2(K) = C_3(K) = \cdots = 0$ .

Hence  $C_0(K) \approx \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$ ,  $C_1(K) \approx \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$ ,  $C_2(K) = C_3(K) = \cdots = 0$ .

Now the sequence (2.1) becomes

$$C_*(K) : 0 \xrightarrow{d_2} C_1(K) \xrightarrow{d_1} C_0(K) \xrightarrow{d_0} 0. \quad (2.2)$$

In order to compute  $H_0(K) = \ker d_0 / \text{im } d_1$ , we need to find  $\ker d_0$  and  $\text{im } d_1$ . First of all it is clear that  $\ker d_0 = C_0(K)$ . Using the definition of  $d_n$  for  $n = 1$  we have:

$$d_1 \langle a, b \rangle = \langle b \rangle - \langle a \rangle$$

$$d_1 \langle a, c \rangle = \langle c \rangle - \langle a \rangle$$

$$d_1 \langle b, c \rangle = \langle c \rangle - \langle b \rangle.$$

Therefore,  $\langle a \rangle \sim \langle b \rangle$ , since  $d_1 \langle a, b \rangle = \langle b \rangle - \langle a \rangle$ , and

$$\langle b \rangle \sim \langle c \rangle, \text{ since } d_1 \langle b, c \rangle = \langle c \rangle - \langle b \rangle.$$

Therefore  $\langle a \rangle \sim \langle b \rangle \sim \langle c \rangle$ , so  $H_0(K) \approx \mathbb{Z}$  with generator  $[\langle a \rangle]$ .

Now let us compute  $H_1(K) = \ker d_1 / \text{im } d_2$ . Clearly,  $\text{im } d_2 = 0$ . To find  $\ker d_1$ , let  $x \in \ker d_1$  and write  $x = u_1 \langle a, b \rangle + u_2 \langle a, c \rangle + u_3 \langle b, c \rangle$  for integers  $u_1, u_2, u_3$ . We have

$$\begin{aligned}
 d_1(x) &= 0 \\
 \Rightarrow d_1(u_1 \langle a, b \rangle + u_2 \langle a, c \rangle + u_3 \langle b, c \rangle) &= 0 \\
 \Rightarrow u_1 d_1 \langle a, b \rangle + u_2 d_1 \langle a, c \rangle + u_3 d_1 \langle b, c \rangle &= 0 \\
 \Rightarrow u_1 (\langle b \rangle - \langle a \rangle) + u_2 (\langle c \rangle - \langle a \rangle) + u_3 (\langle c \rangle - \langle b \rangle) &= 0 \\
 \Rightarrow \langle a \rangle (-u_1 - u_2) + \langle b \rangle (u_1 - u_3) + \langle c \rangle (u_2 + u_3) &= 0 \\
 \Rightarrow -u_1 - u_2 = u_1 - u_3 = u_2 + u_3 = 0.
 \end{aligned}$$

Therefore,  $u_1 = -u_2$  and  $u_3 = -u_2$ . Therefore,  $x = k \langle a, b \rangle - k \langle a, c \rangle + k \langle b, c \rangle$  for some  $k \in \mathbb{Z}$ , and thus,  $\langle a, b \rangle - \langle a, c \rangle + \langle b, c \rangle$  generates  $\ker d_1$ , so  $\ker d_1 \approx \mathbb{Z}$ . Then  $H_1(K) = \ker d_1 / \text{im } d_2 = \ker d_1 / 0 = \ker d_1 \approx \mathbb{Z}$ , generated by  $\langle a, b \rangle - \langle a, c \rangle + \langle b, c \rangle$ . In conclusion, we have

$$H_n(S^1) = H_n(K) \approx \begin{cases} \mathbb{Z}, & n = 0, 1 \\ 0, & n > 1. \end{cases}$$

The technique used to find the homology groups in this simple example rapidly becomes very tedious for larger or more complicated simplicial complexes. Using linear algebra helps to increase the efficiency of the computation. Indeed, it gives an algorithmic method for these computations that can be implemented on a computer, as discussed in the following section. Before doing this we briefly define the idea of relative homology, which is also useful in simplifying the computations.

Let  $L$  be a subcomplex of a simplicial complex  $K$ .

Then  $C_*(L)$  forms a sub-chain complex of  $C_*(K)$ . Using Definition 2.2.10 we define

the relative homology of  $K$  and  $L$  by the following definition.

**Definition 2.4.6** *Let  $L$  be a subcomplex of a simplicial complex  $K$ . Then the relative homology of  $K$  with respect to  $L$  is  $H_*(K, L) := H_*(C_*(K), C_*(L))$ .*

## 2.5 Integral Smith Normal Form for Matrices

**Definition 2.5.1** *Let  $M_{m,n}(\mathbb{Z})$  denote the set of all matrices of size  $m \times n$  with entries from  $\mathbb{Z}$ . For any  $B \in M_{m,n}(\mathbb{Z})$ , we say  $B$  is in **ISmith normal form** (integral Smith normal form) if*

$$B = \begin{bmatrix} \text{diag}(x_1, \dots, x_k) & 0_{k, n-k} \\ 0_{m-k, k} & 0_{m-k, n-k} \end{bmatrix},$$

where  $x_i | x_{i+1}$  (i.e.  $x_i$  is a divisor of  $x_{i+1}$ ),  $1 \leq i \leq k-1$ , and  $\text{diag}(x_1, \dots, x_k)$  is the diagonal matrix with  $x_1, \dots, x_k$  on the main diagonal and zeroes elsewhere.

Note: Any  $m \times n$  matrix  $A$  over the integers  $\mathbb{Z}$  has an ISmith normal form  $B$  and it is unique up to multiplication of each  $x_i$  by  $\pm 1$  ([6] p.109, Theorem 7.10 and p.118, Theorem 7.17). The ISmith normal form can be obtained by methods similar to Gaussian elimination but using both row and column operations, and restricted to only integral operations (i.e. division is not allowed). Examples are given in Example 2.5.3 and later.

**Remark 2.5.2** A standard interpretation of a matrix  $A$  over the integers  $\mathbb{Z}$  is as a linear transformation  $\phi : F_n \rightarrow F_m$ , where  $F_t$  is the free abelian group on  $t$  generators. The ISmith form  $B$  for  $A$  can then be interpreted as the matrix of  $\phi$  relative to some

other bases for  $F_n, F_m$ . Since  $B$  has a very simple form, it makes it quite easy to determine  $\text{im } \phi$  and  $\text{ker } \phi$ , and herein lies the advantage of this method.

**Example 2.5.3** Compute the ISmith normal form of the matrix  $A$ , where

$$A = \begin{bmatrix} 4 & -2 & -6 & -4 \\ 14 & 5 & 12 & -14 \\ 5 & -1 & 6 & -5 \end{bmatrix}.$$

$$\begin{array}{ccc} \begin{bmatrix} 4 & -2 & -6 & -4 \\ 14 & 5 & 12 & -14 \\ 5 & -1 & 6 & -5 \end{bmatrix} & \xrightarrow{-C_2} & \begin{bmatrix} 4 & 2 & -6 & -4 \\ 14 & -5 & 12 & -14 \\ 5 & 1 & 6 & -5 \end{bmatrix} \\ \\ R_1 \leftrightarrow R_3 & \xrightarrow{C_1 \leftrightarrow C_2} & \begin{bmatrix} 5 & 1 & 6 & -5 \\ 14 & -5 & 12 & -14 \\ 4 & 2 & -6 & -4 \end{bmatrix} \\ \\ \begin{array}{l} 5R_1 + R_2 \\ -2R_1 + R_3 \end{array} & \xrightarrow{\begin{array}{l} -5C_1 + C_2 \\ -6C_1 + C_3 \\ 5C_1 + C_4 \end{array}} & \begin{bmatrix} 1 & 5 & 6 & -5 \\ 0 & 39 & 42 & -39 \\ 0 & -6 & -18 & 6 \end{bmatrix} \\ \\ C_2 + C_4 & \xrightarrow{6R_3 + R_2} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 39 & 42 & 0 \\ 0 & -6 & -18 & 0 \end{bmatrix} \end{array}$$

$$\begin{array}{ccc}
 2R_2 + R_3 & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & -66 & 0 \\ 0 & 0 & -150 & 0 \end{bmatrix} & \xrightarrow{22C_2 + C_3} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & -150 & 0 \end{bmatrix} = B.
 \end{array}$$

Notice that  $1|3$  and  $3|150$ . Also, the notation  $R_i \leftrightarrow R_j$  denotes the elementary operation interchanging the  $i$ th and  $j$ th rows, while the notation  $aR_i + R_j$  denotes the elementary operation replacing the  $j$ th row by itself plus  $a$  times the  $i$ th row (similarly for columns).

**Notation 2.5.4** It is convenient to use the notation  $A \equiv B$ , when (as in Example 2.5.3)  $B$  is the integral Smith normal form of  $A$ .

**Example 2.5.5** Now let us look at Example 2.4.5 (on page 24) again. We have the sequence:

$$0 \xrightarrow{d_2} C_1(K) \xrightarrow{d_1} C_0(K) \xrightarrow{d_0} 0.$$

Clearly,  $\ker d_0 = C_0(K)$ . In order to find  $H_0(K)$  we need to set up the differential matrix. The differential matrix is constructed as follows: the rows represent the elements of  $C_1(K)$  and the columns represent the elements of  $C_0(K)$ . We have

$$d_1\langle a, b \rangle = \langle b \rangle - \langle a \rangle$$

$$d_1\langle a, c \rangle = \langle c \rangle - \langle a \rangle$$

$$d_1\langle b, c \rangle = \langle c \rangle - \langle b \rangle.$$

Thus the following array represents the homomorphism  $d_1$ :

$d_1$	$\langle a \rangle$	$\langle b \rangle$	$\langle c \rangle$
$\langle a, b \rangle$	-1	1	0
$\langle a, c \rangle$	-1	0	1
$\langle b, c \rangle$	0	-1	1

Now we can form the differential matrix for  $d_1$ , called  $D_1$ . Therefore

$$D_1 = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix}.$$

One readily finds that the ISmith normal form of  $D_1$  is

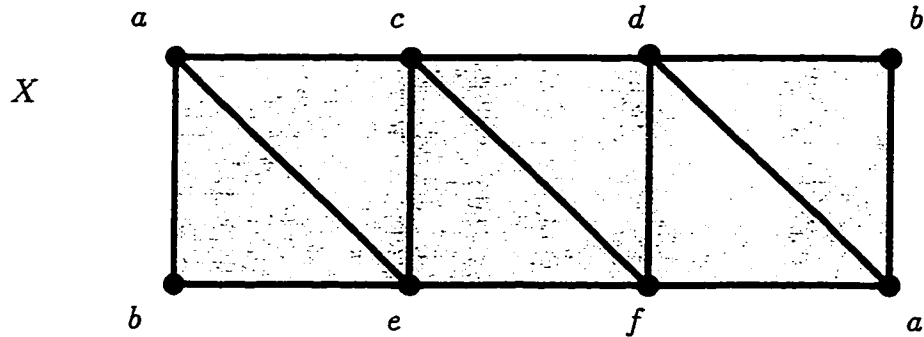
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Having this ISmith form for  $D_1$ , and considering Remark 2.5.2, this means that there are new bases  $\{\alpha_1, \beta_1, \gamma_1\}$  and  $\{\alpha_0, \beta_0, \gamma_0\}$  for  $C_1(K), C_0(K)$  respectively such that  $d_1(\alpha_1) = \alpha_0$ ,  $d_1(\beta_1) = \beta_0$ ,  $d_1(\gamma_1) = 0$ . Thus  $\text{im } d_1$  is generated by  $\alpha_0, \beta_0$ . Since  $\ker d_0 = C_0(K)$ , this implies  $H_0(K) \approx \mathbb{Z}$ , with generator  $[\gamma_0]$ . Furthermore  $\ker d_1$  is generated by  $\gamma_1$ , hence  $H_1(K) = \ker d_1 / \text{im } d_2 = \ker d_1 / 0 = \ker d_1 \approx \mathbb{Z}$ , generated by  $[\gamma_1]$ .

Clearly,  $H_2(K) = H_3(K) = \dots = 0$ .

$$\text{Therefore } H_n(S^1) \approx H_n(K) \approx \begin{cases} \mathbb{Z}, & n = 0, 1 \\ 0, & n > 1. \end{cases}$$

**Example 2.5.6** Let  $X$  be the simplicial complex below. Determine the homology of  $X$ . Notice that the geometrical realization of  $X$  is the Möbius band (left and right vertical edges are identified with opposite orientation).



Let  $L = (V, \Sigma)$  be the abstract simplicial complex of  $X$  with the order  $a < b < c < d < e < f$ . Therefore,  $\Sigma = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}, \{c, f\}, \{d, f\}, \{e, f\}, \{a, b, e\}, \{a, c, e\}, \{a, b, d\}, \{a, d, f\}, \{c, d, f\}, \{c, e, f\}\}$ .

Therefore,

a basis of  $C_0(L)$  is  $\{\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle, \langle f \rangle\}$ ,

a basis of  $C_1(L)$  is  $\{\langle a, b \rangle, \langle a, c \rangle, \langle a, d \rangle, \langle a, e \rangle, \langle a, f \rangle, \langle b, d \rangle, \langle b, e \rangle, \langle c, d \rangle, \langle c, e \rangle, \langle c, f \rangle, \langle d, f \rangle, \langle e, f \rangle\}$ ,

a basis of  $C_2(L)$  is  $\{\langle a, b, e \rangle, \langle a, c, e \rangle, \langle a, b, d \rangle, \langle a, d, f \rangle, \langle c, d, f \rangle, \langle c, e, f \rangle\}$ ,

and  $C_3(L) = C_4(L) = \dots = 0$ .

Thus we have the following sequence

$$C_*(L) : 0 \xrightarrow{d_3} C_2(L) \xrightarrow{d_2} C_1(L) \xrightarrow{d_1} C_0(L) \xrightarrow{d_0} 0.$$

Now let us construct the differential matrices, as shown in the following arrays.

$d_2$	$\langle a, b \rangle$	$\langle a, c \rangle$	$\langle a, d \rangle$	$\langle a, e \rangle$	$\langle a, f \rangle$	$\langle b, d \rangle$	$\langle b, e \rangle$	$\langle c, d \rangle$	$\langle c, e \rangle$	$\langle c, f \rangle$	$\langle d, f \rangle$	$\langle e, f \rangle$
$\langle a, b, e \rangle$	1	0	0	-1	0	0	1	0	0	0	0	0
$\langle a, b, d \rangle$	1	0	-1	0	0	1	0	0	0	0	0	0
$\langle a, c, e \rangle$	0	1	0	-1	0	0	0	0	1	0	0	0
$\langle a, d, f \rangle$	0	0	1	0	-1	0	0	0	0	0	1	0
$\langle c, d, f \rangle$	0	0	0	0	0	0	0	1	0	-1	1	0
$\langle c, e, f \rangle$	0	0	0	0	0	0	0	0	1	-1	0	1

$d_1$	$\langle a \rangle$	$\langle b \rangle$	$\langle c \rangle$	$\langle d \rangle$	$\langle e \rangle$	$\langle f \rangle$
$\langle a, b \rangle$	-1	1	0	0	0	0
$\langle a, c \rangle$	-1	0	1	0	0	0
$\langle a, d \rangle$	-1	0	0	1	0	0
$\langle a, e \rangle$	-1	0	0	0	1	0
$\langle a, f \rangle$	-1	0	0	0	0	1
$\langle b, d \rangle$	0	-1	0	1	0	0
$\langle b, e \rangle$	0	-1	0	0	1	0
$\langle c, d \rangle$	0	0	-1	1	0	0
$\langle c, e \rangle$	0	0	-1	0	1	0
$\langle c, f \rangle$	0	0	-1	0	0	1
$\langle d, f \rangle$	0	0	0	-1	0	1
$\langle e, f \rangle$	0	0	0	0	-1	1

Thus we have the corresponding matrices,



$$D_2 = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}$$

$$D_1 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

After computing the ISmith normal forms of  $D_2$  and  $D_1$  we obtain

$$D_2 \equiv \begin{bmatrix} I_6 & 0_{6 \times 6} \end{bmatrix} \text{ and } D_1 \equiv \begin{bmatrix} I_5 & 0_{5 \times 1} \\ 0_{7 \times 5} & 0_{7 \times 1} \end{bmatrix},$$

where  $I_n$  represents the  $n \times n$  identity matrix and  $0_{m \times n}$  is the  $m$  by  $n$  zero matrix.

Therefore,

$$\ker d_2 = 0, \quad \text{im } d_2 \approx \mathbb{Z}^6,$$

$$\ker d_1 \approx \mathbb{Z}^7, \quad \text{im } d_1 \approx \mathbb{Z}^5.$$

It follows that

$$H_2(L) = \ker d_2 / \text{im } d_3 = 0,$$

$$H_1(L) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^7 / (\mathbb{Z}^6 \oplus 0) \approx \mathbb{Z},$$

$$H_0(L) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^6 / (\mathbb{Z}^5 \oplus 0) \approx \mathbb{Z}.$$

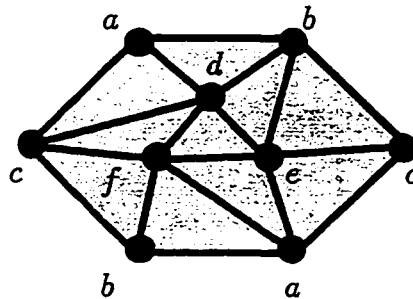
We also know that  $H_i(L) = 0$  for all  $i > 2$  since  $C_i(L) = 0$  for all  $i > 2$ .

Therefore,

$$H_i(X) \approx H_i(L) \approx \begin{cases} \mathbb{Z}, & \text{for } i = 0, 1 \\ 0, & \text{for } i \geq 2. \end{cases}$$

Notice that the homology of the Möbius band is the same as the homology of a circle, which was computed in Example 2.4.5. There is a reason for this, namely the two spaces have the same homotopy type (see 4.2.7; this is a more general equivalence than homeomorphism), and this suffices to imply their homology groups are isomorphic.

**Example 2.5.7** Let  $X$  be the simplicial complex below. Determine the homology of  $X$  (its geometrical realization is the real projective plane  $\mathbb{R}P^2$ , which can be thought of as a disc with antipodal points on the boundary identified).



Let  $L = (V, \Sigma)$  be the abstract simplicial complex of  $X$ , with the vertex order  $a < b < c < d < e < f$ . Without going through the tedious calculations, we will list  $\Sigma$  and state the homology. Thus,  $\Sigma = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{b, c\}, \{b, d\}, \{b, e\}, \{b, f\}, \{c, d\}, \{c, e\}, \{c, f\}, \{d, e\}, \{d, f\}, \{e, f\}, \{a, b, d\}, \{a, b, f\}, \{a, c, e\}, \{a, c, d\}, \{a, e, f\}, \{b, c, e\}, \{b, c, f\}, \{b, d, e\}, \{c, d, f\}, \{d, e, f\}\}$ .

Now we will determine a basis for  $C_n(L)$  for  $n \geq 0$ .

A basis for  $C_0(L)$  is  $\{\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle, \langle f \rangle\}$ .

A basis for  $C_1(L)$  is  $\{\langle a, b \rangle, \langle a, c \rangle, \langle a, d \rangle, \langle a, e \rangle, \langle a, f \rangle, \langle b, c \rangle, \langle b, d \rangle, \langle b, e \rangle, \langle b, f \rangle, \langle c, d \rangle, \langle c, e \rangle, \langle c, f \rangle, \langle d, e \rangle, \langle d, f \rangle, \langle e, f \rangle\}$ .

A basis for  $C_2(L)$  is  $\{\langle a, b, d \rangle, \langle a, b, f \rangle, \langle a, c, d \rangle, \langle a, c, e \rangle, \langle a, e, f \rangle, \langle b, c, e \rangle, \langle b, c, f \rangle, \langle b, d, e \rangle, \langle c, d, f \rangle, \langle d, e, f \rangle\}$ .

Finally,  $C_n(L) = 0$  for all  $n \geq 3$ .

Thus we consider the sequence

$$C_n(L) : 0 \xrightarrow{d_3} C_2(L) \xrightarrow{d_2} C_1(L) \xrightarrow{d_1} C_0(L) \xrightarrow{d_0} 0.$$

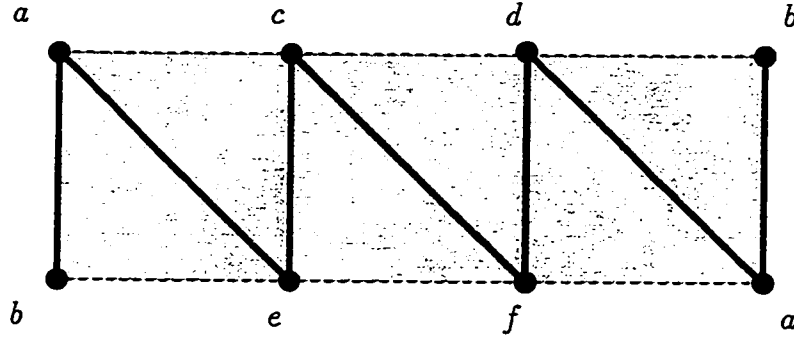
The homology of  $L$  is

$$H_n(\mathbb{R}P^2) \approx H_n(L) \approx \begin{cases} \mathbb{Z}, & n = 0, \\ \mathbb{Z}_2 & n = 1, \\ 0, & n > 1. \end{cases}$$

We now turn to an example of relative homology.

**Example 2.5.8** Let us consider the Möbius band  $X$  as in Example 2.5.6 and let  $Y$  be the subcomplex consisting of its boundary. Define  $M$  to be the subcomplex of

$L$  corresponding to  $Y$ , its maximal simplexes are  $\{a, c\}$ ,  $\{c, d\}$ ,  $\{b, d\}$ ,  $\{a, f\}$ ,  $\{b, e\}$ ,  $\{e, f\}$  (the dashed lines in the diagram).



Therefore,

a basis of  $C_0(M)$  is  $\{\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle, \langle f \rangle\}$ ,

a basis of  $C_1(M)$  is  $\{\langle a, c \rangle, \langle c, d \rangle, \langle b, d \rangle, \langle a, f \rangle, \langle b, e \rangle, \langle e, f \rangle\}$ ,

a basis of  $C_i(M)$  is  $\emptyset$  for all  $i \geq 2$ , i.e.  $C_i(M) = 0$  for  $i \geq 2$ .

Keeping in mind the bases of  $L$  shown in Example 2.5.6, the relative chains  $C_*(L, M) = C_*(L)/C_*(M)$  are now seen to be

$$C_0(L, M) = 0,$$

$$C_1(L, M) \text{ has basis } \{\langle a, b \rangle, \langle a, e \rangle, \langle a, d \rangle, \langle c, e \rangle, \langle c, f \rangle, \langle d, f \rangle\}$$

$$C_2(L, M) \text{ has basis } \{\langle a, b, e \rangle, \langle a, b, d \rangle, \langle a, c, e \rangle, \langle a, d, f \rangle, \langle c, d, f \rangle, \langle c, e, f \rangle\}, \quad \text{since}$$

$$C_2(L, M) = (C_2(L))/C_2(M) = C_2(L)/0 = C_2(L)$$

$$C_i(L, M) = 0 \quad \text{for all } i \geq 3.$$

Thus we only need to consider the sequence

$$C_*(L, M) : 0 \rightarrow C_2(L, M) \xrightarrow{\tilde{d}_2} C_1(L, M) \xrightarrow{\tilde{d}_1} C_0(L, M) = 0 \xrightarrow{\tilde{d}_0} 0.$$

Therefore, we have the following relationship,

$\tilde{d}_2$	$\langle a, b \rangle$	$\langle a, d \rangle$	$\langle a, e \rangle$	$\langle c, e \rangle$	$\langle c, f \rangle$	$\langle d, f \rangle$
$\langle a, b, e \rangle$	1	0	-1	0	0	0
$\langle a, b, d \rangle$	1	-1	0	0	0	0
$\langle a, c, e \rangle$	0	0	-1	1	0	0
$\langle a, d, f \rangle$	0	1	0	0	0	1
$\langle c, d, f \rangle$	0	0	0	0	-1	1
$\langle c, e, f \rangle$	0	0	0	1	-1	0

(2.3)

Also notice that  $\tilde{d}_1$  maps to 0, which implies that  $\ker \tilde{d}_1 = C_1(L, M)$  and  $\text{im } \tilde{d}_1 = 0$ . Thus we only need to determine  $\ker \tilde{d}_2$  and  $\text{im } \tilde{d}_2$ . To do this we will resort to the computation of the ISmith normal form of the matrix  $\tilde{D}_2$  which corresponds to the homomorphism  $\tilde{d}_2$  (cf. Array 2.3)

$$\tilde{D}_2 = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

Thus we find that

$$\ker \tilde{d}_2 = 0, \quad \text{im } \tilde{d}_2 \approx \mathbb{Z}^5 \oplus 2\mathbb{Z},$$

$$\ker \tilde{d}_1 \approx \mathbb{Z}^6, \quad \text{im } \tilde{d}_1 = 0.$$

Therefore,

$$H_2(L, M) = \ker \tilde{d}_2 / \text{im } \tilde{d}_3 = 0,$$

$$H_1(L, M) = \ker \tilde{d}_1 / \text{im } \tilde{d}_2 \approx \mathbb{Z}^6 / (\mathbb{Z}^5 \oplus 2\mathbb{Z}) \approx \mathbb{Z}_2,$$

$$H_0(L, M) = \ker \tilde{d}_0 / \text{im } \tilde{d}_1 = 0.$$

Also since  $C_i(L, M) = 0$  for all  $i > 2$ , this implies that  $H_i(L, M) = 0$  for all  $i > 2$ .

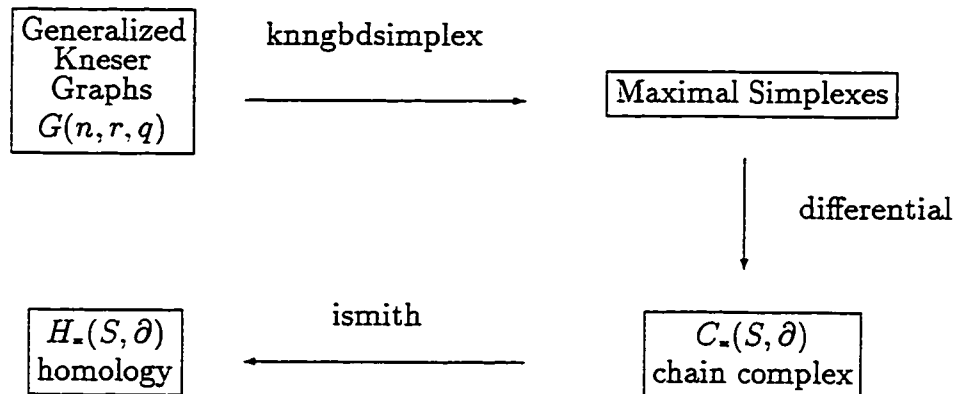
As we have seen, the homology of the Möbius band (Example 2.5.6) is the same as that of a circle, only detecting the 1-dimensional hole and ignoring the twist in the Möbius band. But the relative homology of the Möbius band with respect to its boundary (Example 2.5.8) detects the twist, namely  $H_1(L, M) \approx \mathbb{Z}_2$ .

## 2.6 Computer Algorithm for Homology

There are two main programs, one of which calculates the homology of the neighbourhood complex while the other calculates the relative homology of the neighbourhood complex with respect to the star (cf. 5.4.2) of a vertex. The birth of the latter program was due to the overwhelming size of the differential matrices in the initial program. Both programs generate the edges of the Kneser graph  $G(n, r, q)$  (cf. Chapter 3 for definition) using the **NOEDGES** and the **kneser** procedures. Some idea of the complexity of these calculations may be obtained, for example, from  $G(7, 2, 0)$ . In this case the size of the matrices range up to  $4340 \times 4872$ . For details of the programs see Appendices A and B. We give just an outline here.

The edges are sent to the **knngbdsimplex** procedure, which then generates the maximal simplexes of the neighbourhood complex (see 4.3.7 for definition) of the Kneser graph. The maximal simplexes are then passed over to the **knhomology**

procedure. Both programs generate “backup” file(s) for each differential matrix. This is a safety precaution, just in case the computer crashes at a particular stage resulting in the loss of hours/weeks of computations. If this occurs then the remainder of the computations may be carried out manually, asking MAPLE to run each procedure in the correct order. The file pertaining to the  $i$ th differential matrix  $D_i$  in both programs is called `differentiali`. This `differentiali` file contains the  $i$ th differential matrix in the appropriate form to use Arne Storjohann’s `ISmith` program. Currently his program has not been officially documented (cf. Appendix C). Then, once out of MAPLE, we can use Arne Storjohann’s `ISmith` program, which has been written in C, to compute the integral Smith normal form of the differential matrices. Then we can directly interpret the kernel and image of the associated differential maps and compute the homology groups accordingly.



We close this chapter with three diagrams that may help in the visualization of some of the concepts that have been discussed.


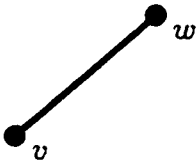
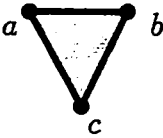
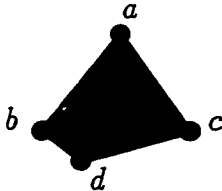
Simplex	Abstract Simplicial Complex	Geometric Realization
(-1)-simplex the empty set	$\{\emptyset\}$	
0-simplex is a vertex.	$\{\emptyset, \{v\}\}$	
1-simplex is an edge.	$\{\emptyset, \{v\}, \{w\}, \{v, w\}\}$	
2-simplex is a (solid) triangle	$\{\emptyset, \{a\}, \{b\}, \{c\},$ $\{a, b\}, \{a, c\}, \{b, c\},$ $\{a, b, c\}\}$	
3-simplex is a (solid) tetrahedron	$\{\emptyset, \{a\}, \{b\}, \{c\}, \{d\},$ $\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\},$ $\{b, d\}, \{c, d\},$ $\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{a, b, d\}$ $\{a, b, c, d\}\}$	

Figure 2.9: Simplexes



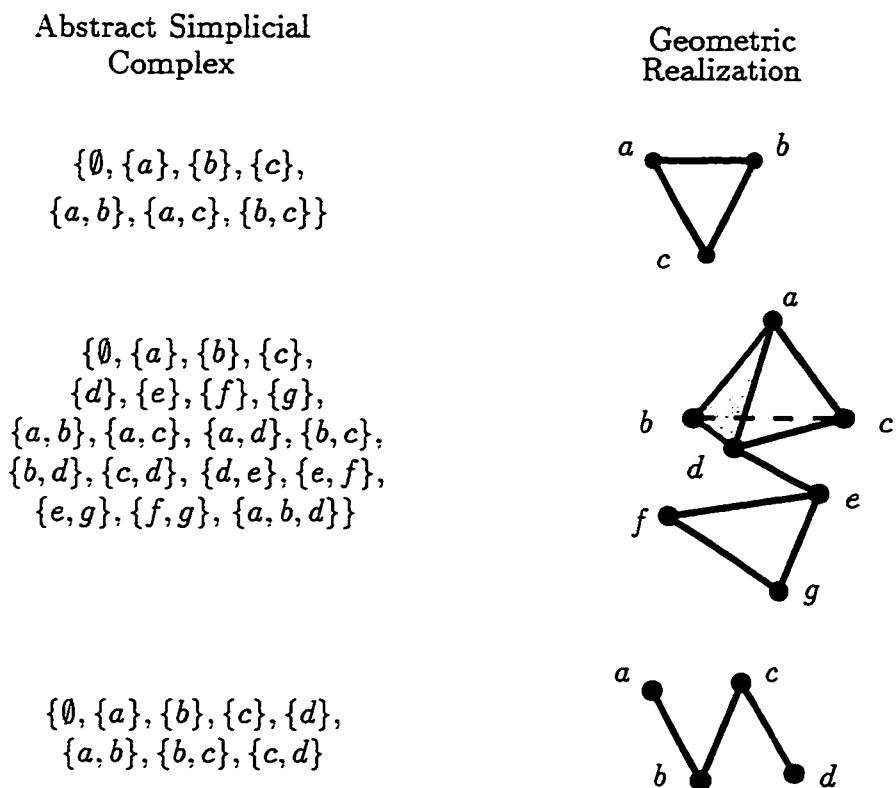


Figure 2.10: Simplicial complexes

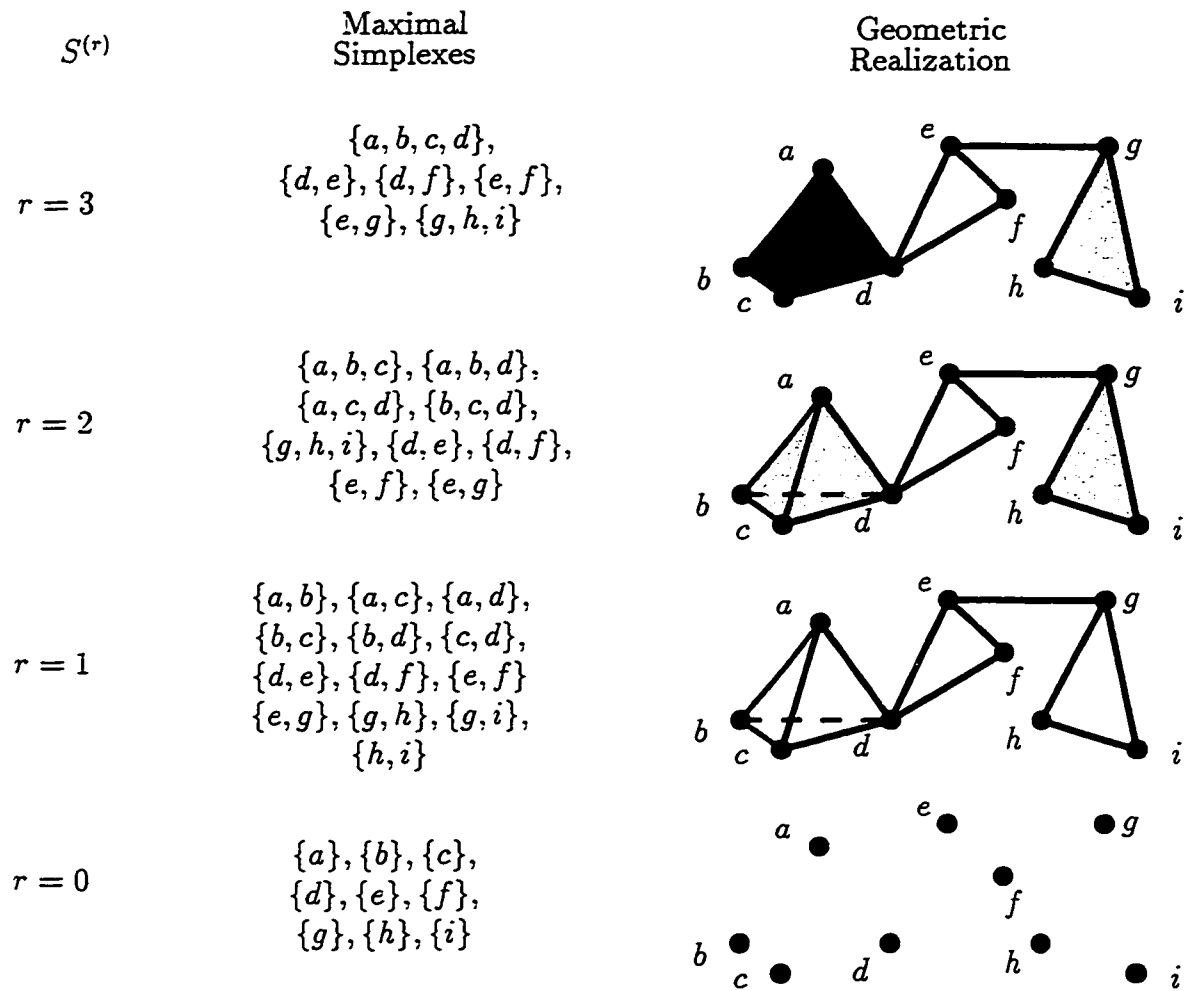


Figure 2.11: Skeleton example.

## Chapter 3

# Graph Colouring

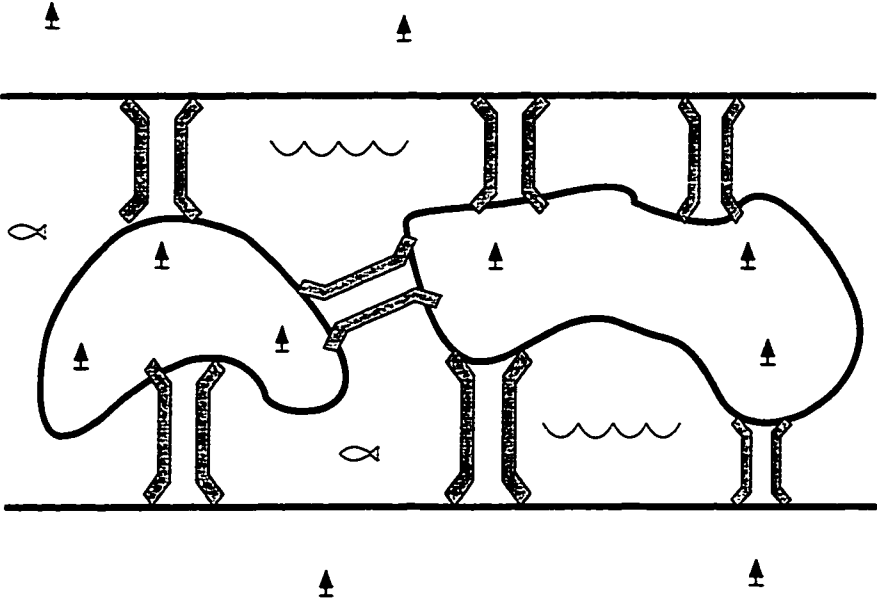
### 3.1 Introduction

There is a very natural link between graph theory and topology. In fact, one could regard graph theory and topology as fraternal twins. Both basically originated with the same paper written by Leonhard Euler in 1736, solving the Königsberg bridge problem (Figure 3.1, page 44). This problem originated with the townspeople of Königsberg, a town that was divided into four regions by the river flowing through it. The question was whether one could find a walk over all seven bridges connecting the various regions, without crossing the same bridge twice and returning to one's starting point.

In a problem of this type (electrical networks form a similar example), distances are irrelevant. Only the connections between the regions are significant. This is typical of graph theory, and more generally of topology. One of the most famous problems in graph theory, and indeed of mathematics, is the celebrated Four Colour Problem (cf.[4] p.148-150,174-175). Graph theory is not only concerned with colouring; other important areas of graph theory are Hamilton cycles, convex embeddings, hereditary systems, algorithms, matchings and factors. There are also numerous applications, such as chemical graphs, networks, broadcasting, as well as applications in other branches of mathematics.

In this chapter we will define graphs and their chromatic number. Then the

The map of Königsberg



Two equivalent graphs representing this map

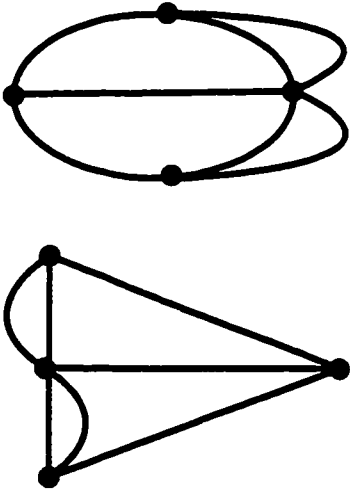


Figure 3.1: Königsberg Bridge Problem

generalized Kneser graphs are introduced. The primary aim of this dissertation is to obtain some information about colouring these graphs, i.e. finding their chromatic number.

## 3.2 Graph Theory and Colourings

Roughly speaking a graph consists of a set of points called vertices and a collection of pairs of these vertices called edges. Graphs (cf. Definition 3.2.2 below) are essentially 1-dimensional simplicial complexes. We start, however, with some more general definitions.

**Definition 3.2.1** *A directed multigraph  $G$  is an ordered pair  $G = (V, E)$ , where*

1.  $V$  is a nonempty set, called the set of vertices,
2.  $E = (e_1, \dots, e_m)$ , where  $e_i \in V \times V$ . The collection  $E$  is called the “set” of edges.

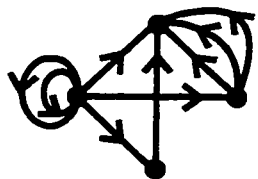
*Note that edges are ordered pairs which may repeat (that is two vertices  $v, w$  may be joined by several edges  $e_i$ ) and loops (i.e. edges of the form  $e = (v, v)$ ) are possible, with this definition of edges.*

*A directed multigraph with no loops and/or multiple edges is called a **directed graph** (or sometimes a **simple directed graph**).*

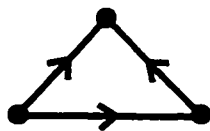
**Definition 3.2.2** *If the edges  $e \in E$  in the graph  $G = (V, E)$  are considered as unordered pairs, i.e.  $(v, w) = (w, v)$ , then  $G$  is called an **undirected multigraph** (or just a **multigraph**). A multigraph with no loops and/or multiple edges is called a **graph** (or a **simple graph**).*

From now on the convention  $(v, w) = (w, v)$  will be assumed for the edges of any undirected multigraph.

### Examples 3.2.3



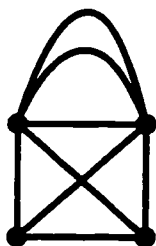
Directed multigraph,  
Loops



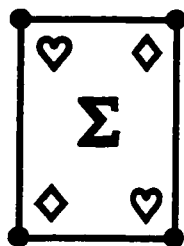
Directed graph,  
No loops



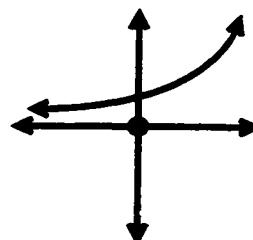
Not a graph



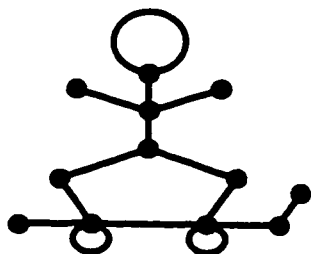
Undirected multigraph,  
No loops



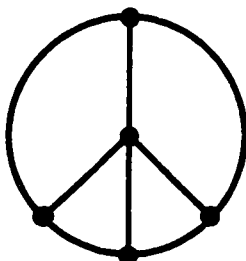
Not a graph



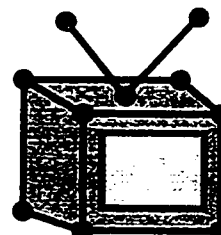
Not a graph



Multigraph,  
Loops



Simple graph



Not a graph

Henceforth we shall deal with only simple graphs.

**Definition 3.2.4** Let  $G = (V, E)$  and  $H = (V', E')$  be graphs.  $H$  is a subgraph of  $G$  if  $V' \subseteq V$  and  $E' \subseteq E$ .

**Definition 3.2.5** A walk in a graph  $G = (V, E)$  from  $a$  to  $b$  ( $a, b \in V$ ) is a sequence of edges  $e_1, e_2, \dots, e_m \in E$  where

$$e_1 = (a, c_1)$$

$$e_2 = (c_1, c_2)$$

$$\vdots$$

$$e_m = (c_{m-1}, b) \text{ with } c_i \in V \text{ for } 1 \leq i \leq m - 1.$$

**Definition 3.2.6** A graph  $G$  is connected if, given any two distinct vertices  $v$  and  $w$ , there exists a walk from  $v$  to  $w$ .

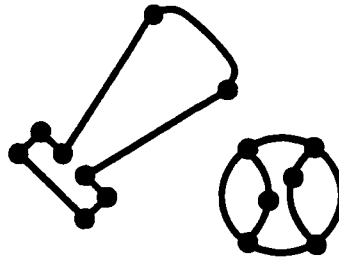
**Remark 3.2.7** According to this definition, a graph with only one vertex and no edges is connected.

**Definition 3.2.8** For any graph  $G$ , a component of  $G$  consists of all edges and vertices which occur in walks starting at some particular vertex of  $G$ . Equivalently, a component is a maximal connected subgraph.

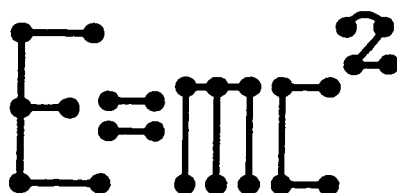
Thus any connected graph has only one component.

### Examples 3.2.9

1. Two components



## 2. Six components



**Definition 3.2.10** The chromatic number,  $\chi(G)$ , of a graph  $G$  is the smallest number of colours required to colour the vertices of a graph so that any two vertices sharing a common edge do not have the same colour.

**Definition 3.2.11** A graph  $G = (V, E)$  is  $k$ -colourable, if  $\chi(G) \leq k$ .

Notice that a  $k$ -colouring of the vertices is equivalent to a partition  $(V_1, \dots, V_k)$  of  $V$  into  $k$  sets, each representing the vertices of one of the  $k$  different colours, such that if  $a, b \in V_i$  then  $(a, b) \notin E$ . Also keep in mind that  $k \leq |V|$ .

**Examples 3.2.12** The reader may like to verify that the following graphs have chromatic numbers, respectively 2,3,4 and 5.

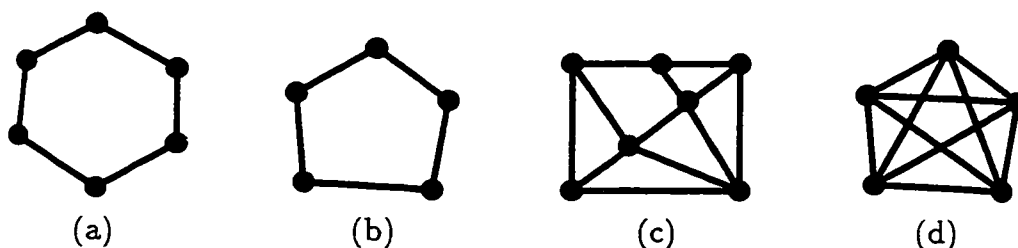


Figure 3.2

**Remark 3.2.13** In Figure 3.2 the graphs (a), (b) are called respectively a 6-cycle ( $C_6$ ) and a 5-cycle ( $C_5$ ); (d) is called a complete graph on five vertices ( $K_5$ ).



**Definition 3.2.14** Let  $G = (V, E)$ ,  $G' = (V', E')$  be two graphs. We say that  $G$  and  $G'$  are isomorphic ( $G \approx G'$ ) if and only if there exists a bijection  $\phi : V \rightarrow V'$  of the vertices such that  $(v, w) \in E$  if and only if  $(\phi(v), \phi(w)) \in E'$ .

### 3.3 Generalized Kneser graphs

**Definition 3.3.1** A generalized Kneser graph,  $G(n, r, q) = (V, E)$ , is a graph with vertex set,  $V = \{A : |A| = r \text{ with } A \subseteq \{1, 2, \dots, n\}\}$ , and with the edge set,  $E = \{(A, B) : |A \cap B| = q, A, B \in V\}$ . Here  $n \geq r \geq q \geq 0$  and  $r \geq 1$ .

One calls  $V$  the  $r$ -subsets of the  $n$ -element set  $\underline{n} = \{1, 2, \dots, n\}$ . Note that the number of such subsets, i.e. the number of vertices in  $G(n, r, q)$ , equals the binomial coefficient  $\binom{n}{r}$ . The cases where  $q = 0$ , i.e.  $A, B$  disjoint, are the classical Kneser graphs (cf. [7]). The next set of figures illustrate a few generalized Kneser graphs.

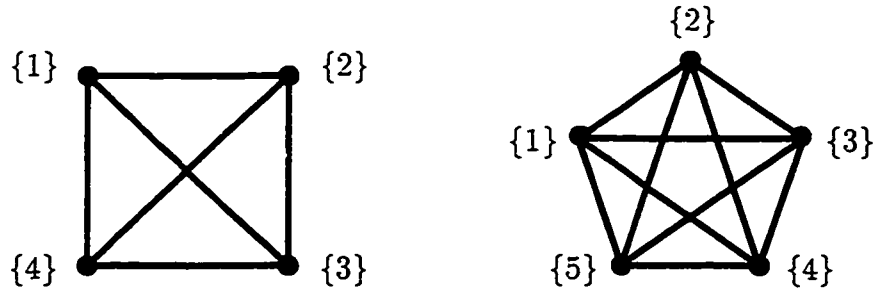
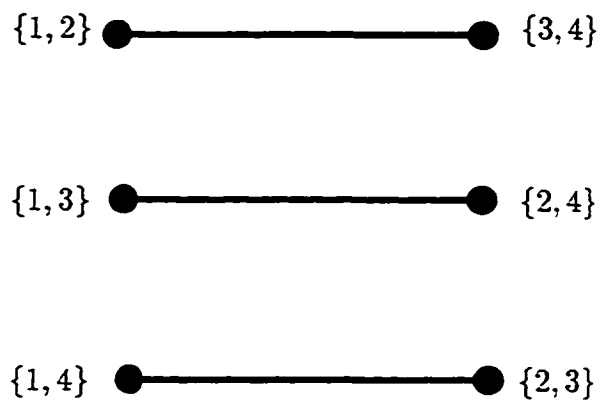
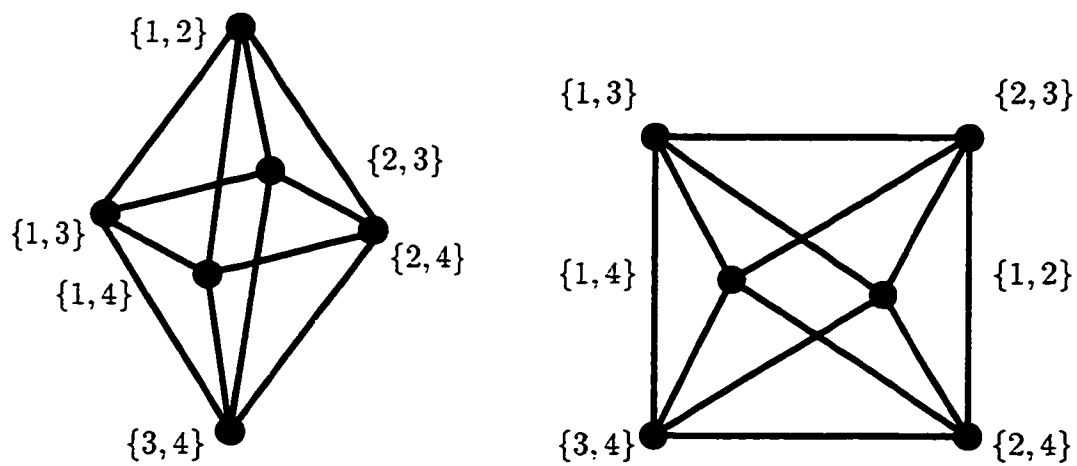
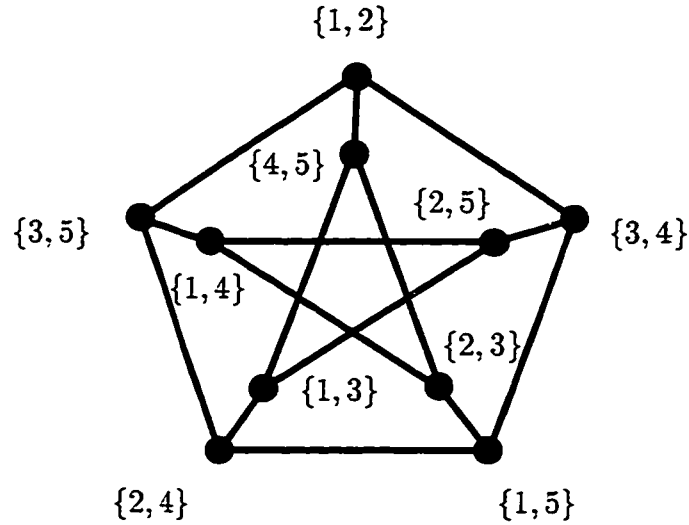


Figure 3.3:  $G(4, 1, 0)$  and  $G(5, 1, 0)$  respectively

Figure 3.4:  $G(4, 2, 0)$ Figure 3.5:  $G(4, 2, 1)$ , two perspectives of the same graph

Figure 3.6:  $G(5, 2, 0)$ 

**Remark 3.3.2** Some of the Kneser graphs are familiar graphs. For example, the graph  $G(4, 2, 1)$  in Figure 3.5 on page 50 is the octahedral graph, and  $G(5, 2, 0)$  in Figure 3.6 is the Petersen graph. Other familiar cases are  $G(n, r, r)$  which is the empty graph on  $\binom{n}{r}$  vertices (that is, a graph with no edges), and  $G(n, 1, 0)$  which is  $K_n$ , the complete graph on  $n$  vertices ( $K_4$  and  $K_5$  are illustrated in Figure 3.3 on page 49).

**Theorem 3.3.3 (Duality)** For  $n \geq r \geq q \geq 0$ ,  $G(n, r, q) \approx G(n, n - r, n - 2r + q)$ .

*Proof:* Let  $G(n, r, q) = (V, E)$  and  $G(n, n - r, n - 2r + q) = (V', E')$ .

Let  $\phi : G(n, r, q) \rightarrow G(n, n - r, n - 2r + q)$  be defined by  $\phi(A) = \underline{n} \setminus A$ . Of course  $|\underline{n} \setminus A| = n - |A| = n - r$ , so  $\phi(A)$  is an  $(n - r)$ -subset of  $\underline{n}$ , i.e. a vertex of  $G(n, n - r, n - 2r + q)$ .

Notice that  $|V| = \binom{n}{r}$  and  $|V'| = \binom{n}{n-r} = \binom{n}{r}$ . Therefore  $|V| = |V'|$ .

First we show that  $\phi$  is one-to-one and onto. To show  $\phi$  is one-to-one, suppose  $\phi(A) = \phi(B)$ , then  $\underline{n} \setminus A = \underline{n} \setminus B$ , and hence  $A = B$ . This implies that  $\phi$  is one-to-

one. Since  $|V| = |V'|$  and is finite,  $\phi$  is must also be onto. Therefore  $\phi$  is a bijection between  $V$  and  $V'$ . Notice that  $\phi^{-1}$  is given by the same formula, i.e.  $\phi^{-1}(C) = \underline{n} \setminus C$ .

We also have to show that  $(A, B) \in E$  if and only if  $(\phi(A), \phi(B)) \in E'$ .

1. Assume that  $(A, B) \in E$ . Then  $|A \cap B| = q$ , and  $|A \cup B| = |A| + |B| - |A \cap B| = 2r - q$ . In order for  $(\phi(A), \phi(B)) \in E'$ ,  $|\phi(A) \cap \phi(B)| = n - 2r + q$ .

$$\begin{aligned}
 |\phi(A) \cap \phi(B)| &= |(\underline{n} \setminus A) \cap (\underline{n} \setminus B)| \\
 &= |\underline{n} \setminus (A \cup B)| \\
 &= n - |A \cup B| \\
 &= n - (2r - q) \\
 &= n - 2r + q.
 \end{aligned}$$

Therefore  $(\phi(A), \phi(B)) \in E'$ .

2. Assume that  $(C, D) \in E'$  for some  $C, D \in V'$ . Then  $|C \cap D| = n - 2r + q$  and  $|C| = |D| = n - r$ . Thus  $|C \cup D| = |C| + |D| - |C \cap D| = 2(n - r) - (n - 2r + q) = n - q$ . In order for  $(\phi^{-1}(C), \phi^{-1}(D)) \in E$  we need  $|\phi^{-1}(C) \cap \phi^{-1}(D)| = q$ .

$$\begin{aligned}
 |\phi^{-1}(C) \cap \phi^{-1}(D)| &= |(\underline{n} \setminus C) \cap (\underline{n} \setminus D)| \\
 &= |\underline{n} \setminus (C \cup D)| \\
 &= n - |C \cup D| \\
 &= n - (n - q) \\
 &= q.
 \end{aligned}$$

Therefore  $(\phi^{-1}(C), \phi^{-1}(D)) \in E$ .

Thus  $(A, B) \in E$  if and only if  $(\phi(A), \phi(B)) \in E'$ , which means that  $\phi$  is an isomorphism. Therefore  $G(n, r, q) \approx G(n, n - r, n - 2r + q)$ .  $\square$

In view of the Duality Theorem, we shall henceforth make the following assumptions for the values of  $n, r$  and  $q$ .

1.  $r > q$ ,
2.  $n + q \geq 2r$ .

The reason for (1) is that, as we have seen in Remark 3.3.2,  $G(n, r, r)$  is an empty graph and therefore of no interest to us. The reason for (2) is that unless  $n + q - 2r \geq 0$  the graph  $G(n, r, q)$  will again be empty; this is a consequence of the the Duality Theorem. Another consequence of the Duality Theorem is that it suffices to just consider those  $G(n, r, q)$  with  $n \geq 2r$ . This is taken into account in the computations, which are thereby reduced by half.

## Chapter 4

# Homotopy and Relations to Graph Colouring

### 4.1 Introduction

The birth of homotopy theory was due to physics and analysis combined with their mutual interest in properties of  $n$ -dimensional manifolds (a circle would be a 1-dimensional manifold, a surface a 2-dimensional manifold, etc.). The first definition of a homotopy group, called the fundamental group ( $\pi_1$ ) was due to Henri Poincaré in 1895. It took about forty years for the higher dimensional homotopy groups ( $\pi_n$ ) to fully emerge into the world. This is usually credited to Witold Hurewicz in 1935, but is actually due to Eduard Čech in 1930. Homotopy theory basically classifies maps from an  $n$ -sphere to a manifold (or more generally to a topological space). It gives rise to very important invariants of topological spaces, namely the homotopy groups. Just like the homology groups in Chapter 2, the homotopy groups have algebraic properties, which help to simplify comparisons and strengthen results or properties of the spaces. In general, the homotopy groups carry more information than the homology groups, but they are more difficult to compute (no algorithm exists, in contrast to homology).

In this chapter we will define homotopy and some of its basic properties. Then we describe connectivity and neighbourhood complexes. We will also show the connection between homotopy and graph colouring which is due to Lovász (see Theorem 4.3.10), along with related theorems and other more recent results.

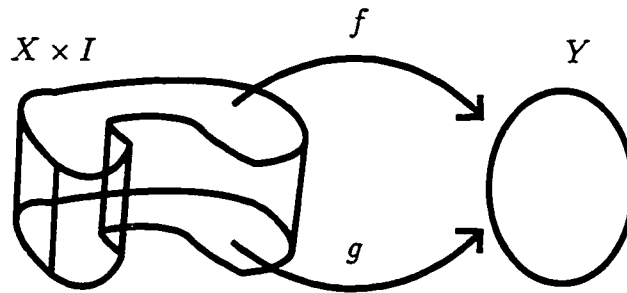
## 4.2 Homotopy Theory

**Definition 4.2.1** Let  $I = [0, 1]$  denote the unit interval. Assume  $f, g : X \rightarrow Y$  are continuous maps from a space  $X$  to space  $Y$ . We say  $f$  is **homotopic** to  $g$ , ( $f \simeq g$ ) if there exists a continuous map  $F : X \times I \rightarrow Y$  such that

$$F(x, 0) = f(x) \text{ and}$$

$$F(x, 1) = g(x) \text{ for all } x \in X.$$

$F$  is called a **homotopy** and we denote this by  $F : f \simeq g$ .



**Definition 4.2.2** Let  $f : X \rightarrow Y$  be a continuous map where  $X$  and  $Y$  are spaces.  $[f] = \{g : X \rightarrow Y, g \text{ continuous and } g \simeq f\}$  is termed the **homotopy class** of  $f$ .

A useful tool for constructing homotopies is the Gluing Lemma.

**Lemma 4.2.3 (Gluing Lemma)** Let  $X = X_1 \cup X_2 \cup \dots \cup X_n$ , where each  $X_i$  is closed in  $X$ . Let  $f_i : X_i \rightarrow Y$  be a map with  $f_i|_{X_i \cap X_j} = f_j|_{X_i \cap X_j}$ , for each  $1 \leq i, j \leq n$ . Then there exists a unique map  $f : X \rightarrow Y$  with  $f|_{X_i} = f_i$ . If each  $f_i$  is continuous, then so is  $f$ .

**Proof:** Clearly, there exists a well defined function  $f$  with  $f|_{X_i} = f_i$ . Now we just have to show that  $f$  is continuous.

Recall that  $f$  is continuous  $\Leftrightarrow f^{-1}(C)$  is closed in  $X$  for all closed  $C \subseteq Y$ .

Let  $C$  be closed in  $Y$ .

$$\begin{aligned}
 f^{-1}(C) &= f^{-1}(C) \cap X \\
 &= f^{-1}(C) \cap (\cup_{i=1}^n X_i) \\
 &= \cup_{i=1}^n (f^{-1}(C) \cap X_i) \\
 &= \cup_{i=1}^n (f_i^{-1}(C)), \text{ since } f|_{X_i} = f_i \text{ for } 1 \leq i \leq n \\
 f_i \text{ is continuous} &\Rightarrow f_i^{-1}(C) \text{ is closed in } X_i \\
 &\Rightarrow f_i^{-1}(C) \text{ is closed in } X \text{ since } X_i \text{ is closed in } X.
 \end{aligned}$$

Hence  $f^{-1}(C)$  is a finite union of closed sets and therefore is itself closed.  $\square$

**Definition 4.2.4** *An equivalence relation  $R$  on a set  $A$  satisfies the following properties:*

1.  $R$  is reflexive, i.e.  $aRa$ , for all  $a \in A$ ,
2.  $R$  is symmetric, i.e.  $aRb \Rightarrow bRa$ , for all  $a, b \in A$  and
3.  $R$  is transitive, i.e.  $aRb$  and  $bRc \Rightarrow aRc$  for all  $a, b, c \in A$ .

### Examples 4.2.5

1.  $=$  is an equivalence relation.
  - (a) Reflexive:  $a = a$ ,
  - (b) Symmetric:  $a = b \Rightarrow b = a$  and
  - (c) Transitive:  $a = b$  and  $b = c \Rightarrow a = c$ .



2.  $<$  is not an equivalence relation.

(a) Reflexivity fails since  $a \not< a$  and

(b) Symmetry fails since  $a < b \not\Rightarrow b < a$ .

**Lemma 4.2.6** *Homotopy is an equivalence relation.*

*Proof:* We need to show that homotopy is reflexive, symmetric and transitive.

1. Show  $\simeq$  is reflexive.

Let  $f : X \rightarrow Y$  be continuous. Let  $J : X \times I \rightarrow Y$  such that  $J(x, t) = f(x)$  for all  $t \in I$ . Therefore,  $J(x, 0) = f(x) = J(x, 1)$ . Since  $f$  is continuous,  $J$  is also continuous. Therefore  $f \simeq f$ .

2. Show that  $\simeq$  is symmetric. Let  $f, g : X \rightarrow Y$  be continuous maps. Let  $F_1 : X \times I \rightarrow Y$  be a continuous map such that  $F_1 : f \simeq g$ . Define  $F_2 : X \times I \rightarrow Y$  such that  $F_2(x, 1 - t) = F_1(x, t)$  for all  $x \in X$ , for all  $t \in I$ .

Clearly  $F_2$  is continuous since  $F_1$  is continuous, the function  $t \mapsto 1 - t$  is continuous, and the composition of continuous functions is continuous. Also

$$F_2(x, 1) = F_1(x, 0) = f(x), \text{ and}$$

$$F_2(x, 0) = F_1(x, 1) = g(x).$$

Therefore,  $F_2 : g \simeq f$ .

3. Show that  $\simeq$  is transitive. Let  $f, g, h : X \rightarrow Y$  be continuous maps. Let  $F_1, F_2 : X \times I \rightarrow Y$ , also be continuous maps such that  $F_1 : f \simeq g$  and

$F_2 : g \simeq h$ . Let us define  $F_3 : X \times I \rightarrow Y$  such that

$$F_3(x, t) = \begin{cases} F_1(x, 2t), & 0 \leq t \leq \frac{1}{2}, \\ F_2(x, 2t - 1), & \frac{1}{2} \leq t \leq 1. \end{cases}$$

Thus,

$$F_3(x, 0) = F_1(x, 0) = f(x) \text{ and}$$

$$F_3(x, 1) = F_2(x, 1) = h(x).$$

Also,

$$F_1(x, 1) = g(x),$$

$$F_2(x, 0) = g(x).$$

Therefore,  $F_3(x, 1/2) = F_1(x, 1) = F_2(x, 0)$  is well defined, and hence  $F_3$  is continuous, by the Gluing Lemma. Thus,  $f \simeq g$  and  $g \simeq h$  implies  $f \simeq h$ .  $\square$

**Definition 4.2.7** *A continuous function  $f : X \rightarrow Y$  is a homotopy equivalence if there exists a continuous function  $g : Y \rightarrow X$  such that  $gf \simeq 1_X$  and  $fg \simeq 1_Y$ .*

In this case the spaces  $X, Y$  are said to be **homotopy equivalent** (or to have the same homotopy type).

**Definition 4.2.8** *A space  $X$  that is homotopy equivalent to a point is called **contractible**.*

**Examples 4.2.9** The spaces  $I, \mathbb{R}, \mathbb{R}^n$ , and any convex subset of  $\mathbb{R}^n$ , are contractible.

For the purposes of homotopy theory, it turns out to be important to consider maps and homotopies that also fix a distinguished point, the “basepoint”, of each space. We now give the minor changes necessary to define these “based homotopies”.

**Definition 4.2.10** Let  $X$  be a topological space and  $x_0 \in X$ . We call  $(X, x_0)$  a **pointed space**;  $x_0$  is called the **basepoint** of  $(X, x_0)$ . A continuous function  $f : (X, x_0) \rightarrow (Y, y_0)$  is a **pointed map** if  $f$  preserves basepoints, i.e.  $f(x_0) = y_0$ .

Notice that if  $(X, x_0)$  is a pointed space then  $X \neq \emptyset$  since  $x_0 \in X$ .

**Definition 4.2.11** Let  $(X, x_0)$  and  $(Y, y_0)$  be pointed spaces, and let  $f, g : (X, x_0) \rightarrow (Y, y_0)$  be pointed maps. We write  $F : f \simeq g \text{ rel } \{x_0\}$  if there is a continuous map  $F : X \times I \rightarrow Y$  with  $F : f \simeq g$  and  $F(x_0, t) = y_0$  for all  $t \in I$ . The homotopy  $F$  is called a **based (pointed) homotopy**.

Since homotopy is an equivalence relation, it is logical to talk about homotopy classes of functions (maps).

**Notation 4.2.12** We will let  $[X, Y]$  denote the set of homotopy classes of maps from  $X$  to  $Y$ ,  $[f]$  denote the homotopy class of  $f$  and  $[X, Y]_*$  denote the set of pointed homotopy classes of pointed maps from pointed spaces  $(X, x_0)$  and  $(Y, y_0)$ .

Using the same convention as Rotman [cf. [11], p.334] we will consider the  $n$ -sphere,  $S^n \subset \mathbb{R}^{n+1}$ , as the set  $\{(a_0, \dots, a_n) : \sum_{i=0}^n a_i^2 = 1\}$ , and  $s_n$  will denote the point  $(1, 0, \dots, 0) \in S^n$  as the basepoint of  $S^n$ .

**Definition 4.2.13** Let  $(X, x_0)$  be a pointed space and let  $n \geq 0$ . We then define  $\pi_n(X, x_0) = [(S^n, s_n), (X, x_0)] = [S^n, X]_*$ , and it is termed the  **$n$ th-homotopy group** of  $(X, x_0)$  (or simply of  $X$ ).

The  $n$ th-homotopy group was first defined for  $n = 1$  by Poincaré [10], and about thirty years later for all  $n \geq 0$  by Čech [3]. One calls  $\pi_n(X, x_0)$  the  $n$ th homotopy

group of  $X$  because of the following theorem, which we do not prove here (cf. [11], p.334-335 Theorem 11.18, p.335 Theorem 11.21).

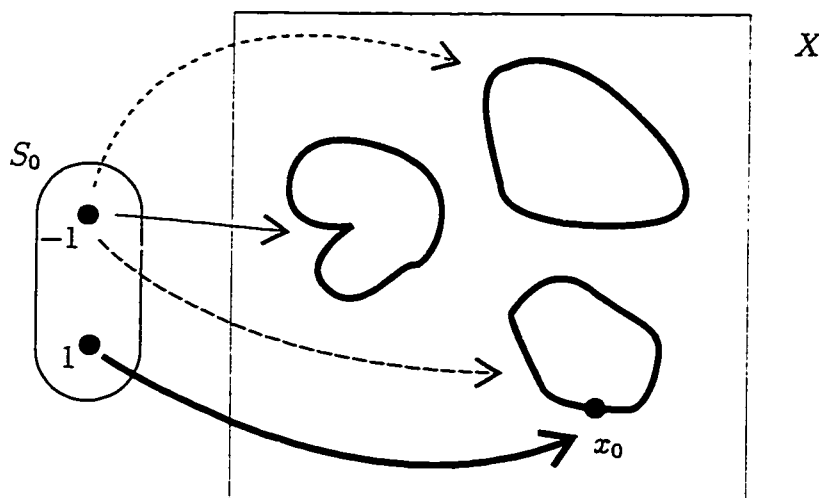
**Theorem 4.2.14** *For  $n \geq 1$ ,  $\pi_n(X, x_0)$  is a group. For  $n \geq 2$ ,  $\pi_n(X, x_0)$  is an abelian group.*

**Definition 4.2.15** *A space  $X$  is path connected if for all  $x, y \in X$ , there exists a continuous map  $f : I \rightarrow X$  such that  $f(0) = x$  and  $f(1) = y$ . The path components of a space  $X$  are its maximal path connected subspaces.*

We conclude this section with three examples.

**Example 4.2.16** There is a 1-1 correspondence between  $\pi_0(X, x_0)$  and the set of path components of  $X$ .

*Informal Proof:* By definition  $\pi_0(X, x_0) = [S^0, X]_* = [\{+1, -1\}, X]_*$ . The basepoint of  $S^0$ , which is 1, must be mapped to the basepoint  $x_0$  of  $X$ . But  $-1$  can be mapped to any element in  $X$ . A map taking  $-1 \mapsto x \in X$  is clearly homotopic to a map taking  $-1 \mapsto y \in X$  if and only if  $x$  and  $y$  can be joined by a path. So the path components of  $X$  are in bijective correspondence with the homotopy classes  $[S^0, X]_* = \pi_0(X, x_0)$ .



**Example 4.2.17** If  $X = \{x_0\}$ , then  $\pi_n(X, x_0) = 0$  for all  $n \geq 0$ .

Since the homotopy groups only depend on the homotopy type of  $X$ , it follows that  $\pi_n(X, x_0) = 0$  for all  $n \geq 0$  when  $X$  is contractible.

**Example 4.2.18**  $\pi_1(S^1, s_1) \approx \mathbb{Z}$ .

No formal proof is supplied, but intuitively any map of  $S^1$  to itself will wind the circle around itself some integer number of times, and this winding number gives the above isomorphism.

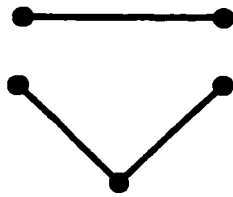
### 4.3 Connectivity and Two Famous Theorems

**Definition 4.3.1** A pointed space  $(X, x_0)$  is said to be  $n$ -connected ( $n \geq 0$ ) if and only if  $\pi_i(X, x_0) = 0$ ,  $0 \leq i \leq n$ . As a convention, if the space  $X$  is not even path connected (0-connected), then we say it is  $(-1)$ -connected.

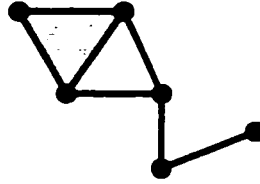
**Notation 4.3.2** If  $X$  is  $n$ -connected but not  $(n+1)$ -connected, write  $\text{conn}(X) = n$ .

### Examples 4.3.3

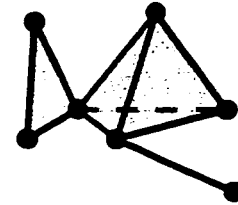
1. A 0-connected space is the same as a path connected space.
2. A 1-connected space is frequently called **simply connected**. In analysis (and calculus) these are thought of as spaces that are connected and in which any closed curve can be shrunk to a point.
3. (Without proof)  $S^n$  is  $(n - 1)$ -connected cf.[12], p.129.



(-1)-connected



0-connected



1-connected

We now state two famous theorems, although the first one is mainly famous to homotopy theorists, and the second to graph theorists.

**Theorem 4.3.4 (Hurewicz Theorem ([11] p.369))** *For a simply connected space  $X$ , and  $n \geq 1$ , the following two conditions are equivalent:*

1.  $H_i(X) = 0$ ,  $1 \leq i \leq n$ ,
2.  $\pi_i(X, x_0) = 0$ ,  $1 \leq i \leq n$ .

*Either (1) or (2) implies that there is an isomorphism  $\pi_{n+1}(X, x_0) \approx H_{n+1}(X)$ , and an epimorphism  $\pi_{n+2}(X, x_0) \twoheadrightarrow H_{n+2}(X)$ .*

**Remark 4.3.5** The map giving this latter isomorphism (or epimorphism) is the Hurewicz homomorphism  $h_q : \pi_q(X, x_0) \rightarrow H_q(X)$ , for any  $q \geq 0$ . In the above

theorem  $H_0(X)$  and  $\pi_0(X)$  were not mentioned, this is because  $H_0(X) \approx \mathbb{Z}$  and  $\pi_0(X, x_0) = 0$  are always true for any path connected space, as is the case here.

Before turning to the second theorem, due to Lovász [7], one definition is needed. We consider henceforth graphs with no isolated vertices, and will seldom mention this rather trivial restriction again.

**Definition 4.3.6** *A topological space  $X$  is a polyhedron if there exists a simplicial complex  $K$  and a homeomorphism  $h : |K| \rightarrow X$ . The ordered pair  $(K, h)$  is called a triangulation of  $X$ .*

**Definition 4.3.7** *Let  $G = (V, E)$  be a graph. The neighbourhood complex, of  $G$  is  $N = N_G = (V_G, \Sigma_G)$  where  $V_G = V$  and  $\Sigma_G = \{\{v_0, v_1, \dots, v_n\} : v_i \in V, \text{ and there exists } w \in V \text{ where } (w, v_i) \in E \text{ for all } 0 \leq i \leq n\}$ .*

**Notation 4.3.8** *We let  $|N_G|$  denote the polyhedron of  $N_G$  and write  $\mathbf{X}_G = |N_G|$ .*

The next few examples should help to give the reader a better intuitive grasp of this concept.

**Examples 4.3.9** Let  $G = K_4 = (V, E)$  be the graph where  $V = \{v_1, v_2, v_3, v_4\}$  and  $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}$ . Thus the neighbours of

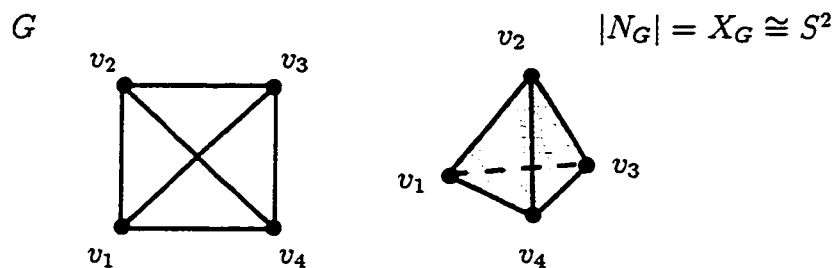
$v_1$  are  $v_2, v_3, v_4,$

$v_2$  are  $v_1, v_3, v_4,$

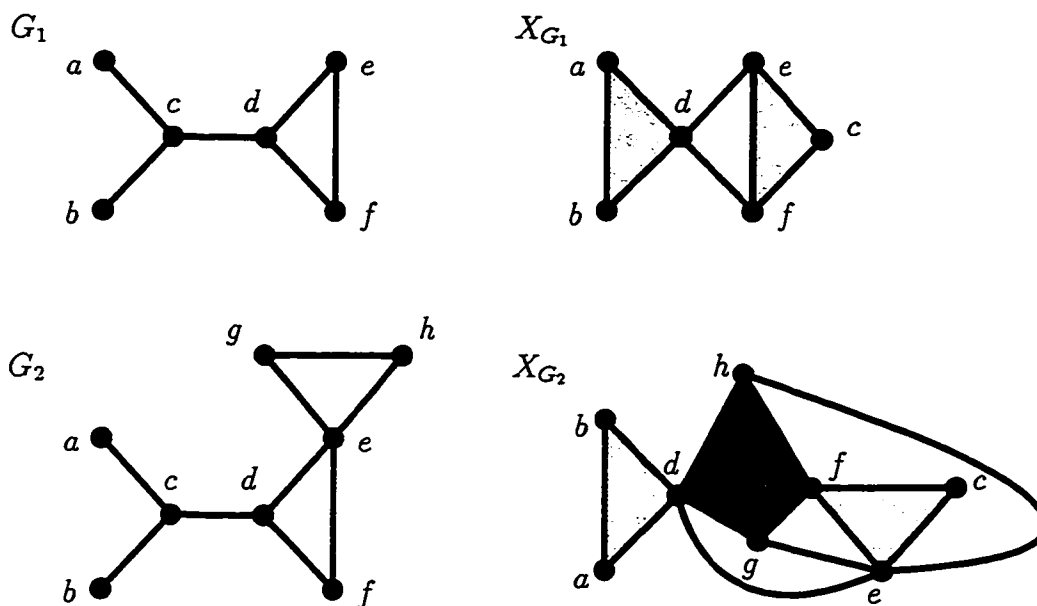
$v_3$  are  $v_1, v_2, v_4,$

$v_4$  are  $v_1, v_2, v_3.$

Therefore  $N_G = \{\{v_2, v_3, v_4\}, \{v_1, v_3, v_4\}, \{v_1, v_2, v_4\}, \{v_1, v_2, v_3\}\}$ . (Here we have listed just the maximal simplexes, recall Definition 2.3.4 and Remark 2.3.5).



Without going through all the details, we illustrate four more examples below.



Notice that in Figure 4.1 there are still further identifications of vertices here (for example the two  $\{1, 3\}$  are the same vertex).

We can now state Lovász's Theorem (recall that we are always considering graphs with no isolated vertices, and do not state this explicitly).

**Theorem 4.3.10 (Lovász ([7]))** *For any graph  $G$ , if  $X_G$  is  $n$ -connected, then  $\chi(G) \geq n + 3$ .*

One can write this theorem as  $\chi(G) \geq \text{conn}(X_G) + 3$ . It is now illustrated with several examples.



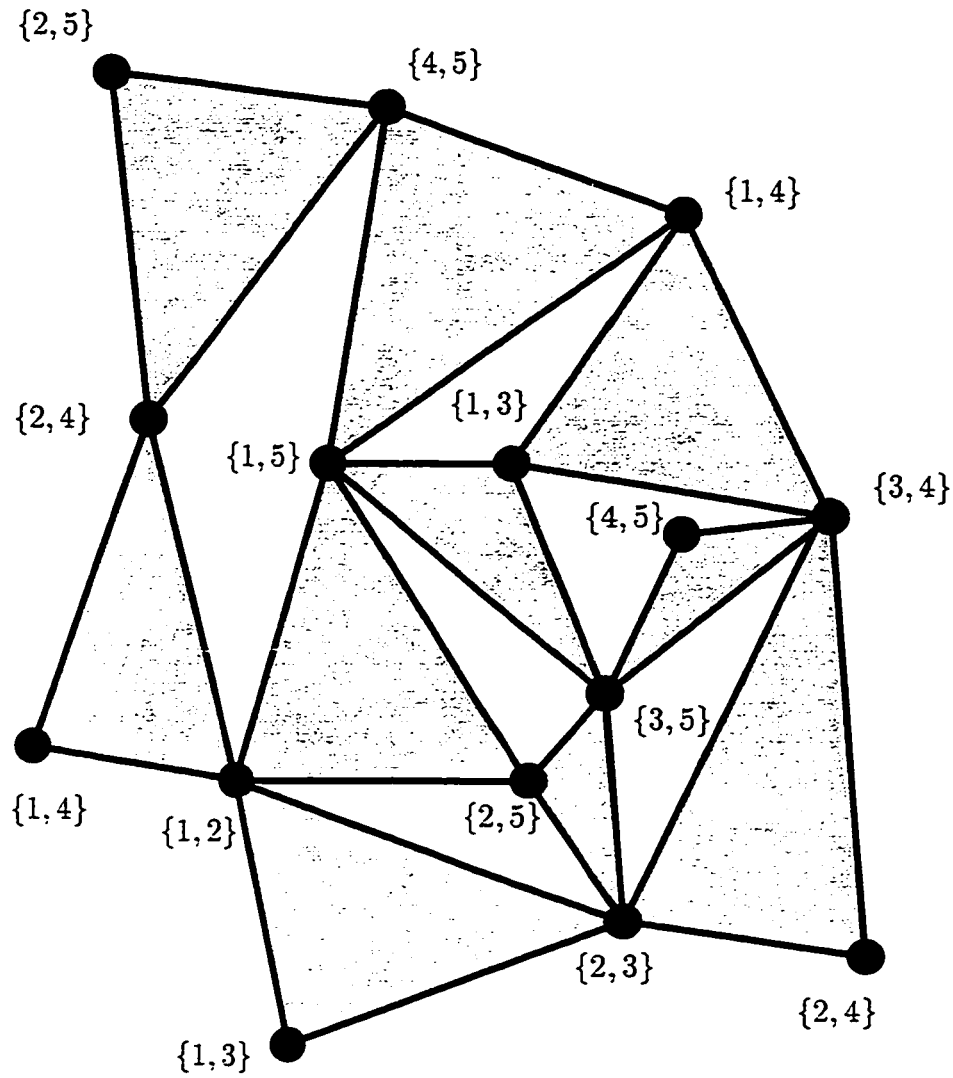


Figure 4.1:  $N(5, 2, 0)$ , see Figure 3.6 for  $G(5, 2, 0)$

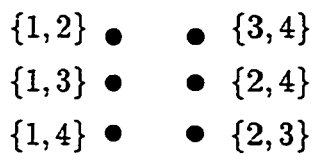
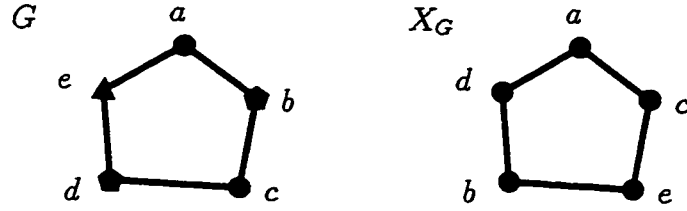
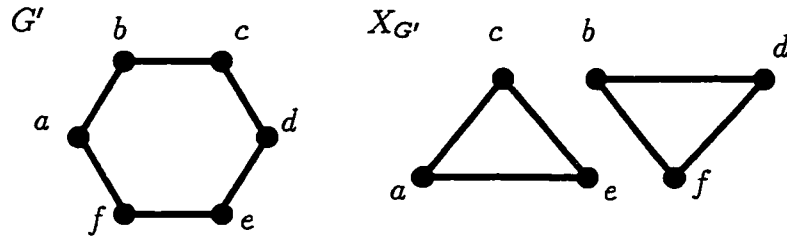


Figure 4.2:  $N(4, 2, 0)$  (see Figure 3.4 for  $G(4, 2, 0)$ )

**Example 4.3.11**

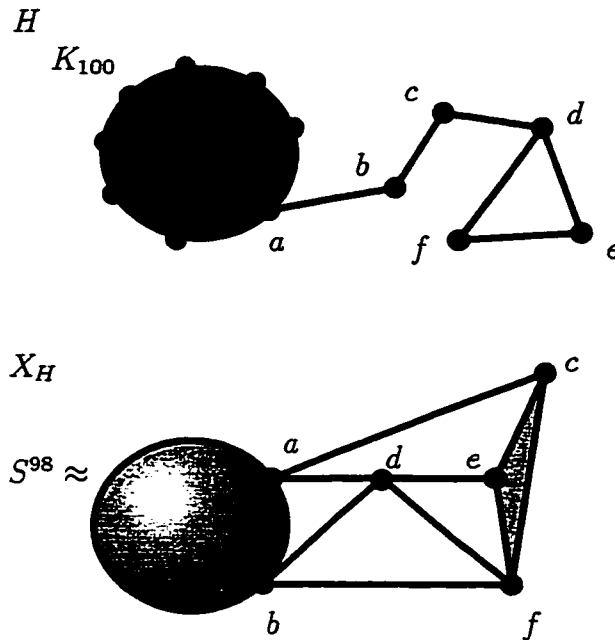
As illustrated by the different shaped vertices, one can easily see that the chromatic number of  $G$  is 3. Now considering  $X_G$ , we can see that  $X_G \approx S^1$  which implies that  $X_G$  is 0-connected, from Examples 4.3.3. This implies that  $\chi(G) \geq 0 + 3 = 3$ .

**Example 4.3.12**

Here  $\chi(G') = 2$  and  $\text{conn}(X_{G'}) = -1$ , so again  $\chi(G') = \text{conn}(X_{G'}) + 3$ .

**Example 4.3.13** A third example is the complete graph on  $n$  vertices,  $K_n$ . It is not hard to see (as in the first example of a neighbourhood complex, 4.3.11), that  $X_{K_n} \cong S^{n-2}$ , which has connectivity  $n - 3$  (cf. 4.3.3). Clearly  $\chi(K_n) = n$ , so once again  $\chi(K_n) = \text{conn}(X_{K_n}) + 3$ .

**Example 4.3.14**



Clearly the chromatic number  $\chi(H)$  in this case is 100, since  $K_{100}$  is part of the graph. But  $X_H$  is only 0-connected due to a 1-dimensional hole  $\{e, f\}, \{e, d\}, \{d, f\}$ , resulting in the following lower bound for  $H$ :  $100 = \chi(H) \geq 0 + 3 = 3$ . This example shows that, in general, the lower bound furnished by Theorem 4.3.10 can be quite weak.

We close this section with another famous theorem in topology, the Borsuk-Ulam Theorem ([11], p.413). Although we do not directly use it in this thesis, it is instrumental in the proofs of both the Lovász Theorem and the generalization described in Section 4.4.

**Theorem 4.3.15 (Borsuk-Ulam Theorem)** *There does not exist any continuous map  $f : S^n \rightarrow S^{n-1}$  for  $n > 0$  such that  $f(-x) = -f(x)$  for all  $x \in S^n$ .*

An equivalent theorem to the Borsuk-Ulam Theorem is the following.

**Theorem 4.3.16** *If  $S^n = \cup_{i=1}^{n+1} A_i$ , where each  $A_i$  is an open (or closed) set of  $S^n$  for all  $1 \leq i \leq n+1$ , then at least one of the sets  $A_i$  contains two antipodal points.*

## 4.4 Recent Developments

In this section we give a brief report of a few developments related to the 1978 paper of Lovász described in Section 4.3. The first is a paper of Alon, Frankel and Lovász [1] in 1986. This work extends the original ideas of Lovász to hypergraphs (a generalization of a graph that we do not define here). Just as the Kneser Conjecture was solved as an application of [7], the second paper [1] is applied to solve an Erdős Conjecture that generalizes the Kneser Conjecture.

Another development is given by the work of Milgram and Zvengrowski in [8]. Using further techniques from homotopy theory, in particular the notions of a principal  $\mathbb{Z}_p$ -bundle for a prime  $p$  and its classifying space, these authors give stronger versions of the Lovász Theorem 4.3.10 and the corresponding theorem in [1]. We do not give the details of their theorem here, but will mention that for Example 4.3.14 their theorem gives the correct lower bound  $\chi(G) \geq 100$  (while the Lovász Theorem gives only  $\chi(G) \geq 3$ ).

## Chapter 5

### Applications to Generalized Kneser Graphs

#### 5.1 Preliminaries

We start with a useful notation.

**Notation 5.1.1** *For any graph  $G = (V, E)$  and any vertex  $v \in V$ , we will write  $\sigma(v) = \{a : a \in V, (a, v) \in E\}$ . Thus  $\sigma(v)$  is the set of all neighbours of  $v$  in  $G$ , and at the same time is a maximal simplex of the neighbourhood complex,  $N_G$ , of  $G$ .*

Secondly, we mention some properties and notation for paths that will be needed in Section 5.3.

By a **path**  $\lambda$  in a topological space  $X$  we mean a continuous map  $\lambda : I \rightarrow X$ . The points  $\lambda(0), \lambda(1)$  are called respectively the **initial** and **terminal** points of the path  $\lambda$ . When  $\lambda(0) = \lambda(1)$  we call  $\lambda$  a **loop**. When one speaks of homotopies of paths, for example  $\lambda \simeq \mu$ , it is always understood that  $\lambda, \mu$  have the same initial and terminal points, and that these remain fixed during the homotopy. Let us give a precise definition.

**Definition 5.1.2** *Two paths  $\lambda, \mu : I \rightarrow X$  are homotopic if there exists a continuous map  $H : I \times I \rightarrow X$  with*

$$H(s, 0) = \lambda(s),$$

$$H(s, 1) = \mu(s),$$

$$H(0, t) = \lambda(0) = \mu(0),$$

$$H(1, t) = \lambda(1) = \mu(1).$$

It should be noted that the terms “path” and “loop” are used in this chapter with the above (topological) meaning. They are used in a slightly different way in graph theory.

Now suppose  $X = |K|$  is the polyhedron of some simplicial complex  $K$ .

**Definition 5.1.3** *A path  $\lambda : I \rightarrow X = |K|$  is simplicial if there exist vertices  $v_0, v_1, \dots, v_n$  of  $X$  such that*

$$\lambda(s) = (1 + i - ns)v_i + (ns - i)v_{i+1}, \quad \frac{i}{n} \leq s \leq \frac{i+1}{n}, \quad 0 \leq i \leq n-1.$$

Intuitively, a simplicial path is a path that travels “linearly” from  $v_0$  to  $v_1$ , then from  $v_1$  to  $v_2$ , etc. It is completely determined by the vertices  $v_0, v_1, \dots, v_n$ , and we usually denote it in exactly this way. Also note that a simplicial path is contained in the 1-skeleton,  $X^{(1)}$ , of the polyhedron  $X$ .

**Theorem 5.1.4** ([12] Section 3.6) *Any path  $\lambda : I \rightarrow X$ , with initial and terminal points which are vertices of  $X$ , is homotopic to a simplicial path.*

It is a well known theorem (cf. [11], p.46 Theorem 3.6) that for a path connected space  $X$ ,  $\pi_1(X, x_0)$  is independent of  $x_0$  up to isomorphism. We therefore have the

following corollary of Theorem 5.1.4, which will be of the utmost importance in Section 5.3.

**Corollary 5.1.5** *For any connected polyhedron  $X = |K|$ ,  $\pi_1(X, x_0)$  can be determined by taking  $x_0$  to be any vertex of  $X$ , and considering only simplicial loops  $v_0, v_1, \dots, v_n$ , where  $x_0 = v_0 = v_n$ .*

Finally, we return to the generalized Kneser graphs  $G(n, r, q)$  and mention some of their basic properties.

Any Kneser graph  $G(n, r, q)$  has many symmetry properties. In particular, each vertex of  $G(n, r, q)$  has  $k = \binom{r}{q} \binom{n-r}{r-q}$  neighbours. In the language of graph theory,  $G(n, r, q)$  is a  $k$ -regular graph. A further symmetry property is given by the next lemma.

**Lemma 5.1.6** *Let  $A, B$  be any two vertices of  $G(n, r, q)$ . Then there exists a graph automorphism,  $\phi$ , of  $G(n, r, q)$  onto itself such that  $\phi(A) = B$ .*

Proof: Let  $A = \{a_1, \dots, a_r\}, B = \{b_1, \dots, b_r\}$  where  $1 \leq a_i, b_i \leq n$  for all  $1 \leq i \leq r$ . There is clearly a permutation  $\tau$  of  $\{1, \dots, n\}$  such that  $\tau(a_i) = b_i$ , for all  $1 \leq i \leq r$  ( $\tau$  is in general not unique, since  $n > r$ ). The vertex map  $\phi : V \rightarrow V$  defined by  $\phi\{c_1, \dots, c_r\} = \{\tau(c_1), \dots, \tau(c_r)\}$  clearly induces the desired automorphism of  $G(n, r, q)$ .  $\square$

Combining Corollary 5.1.5 and Lemma 5.1.6, we see that the computation of  $\pi_1(X_G, x_0)$ , where  $G = G(n, r, q)$ , can be carried out with  $x_0$  any vertex. In particular we shall choose  $x_0 = \{1, 2, \dots, r\} := A_0$ .

## 5.2 0-Connectivity

**Theorem 5.2.1** *Let  $G = G(n, r, q)$ . If  $q \geq 1$  or if  $q = 0$  and  $n \geq 2r + 1$ , then  $X_G$  is path connected.*

*Proof:* Let  $G = (V, E) = G(n, r, q)$ . We need to show that between any two vertices  $A, B$  in  $N = N(n, r, q)$  there exists a path. It suffices to show that  $\{A, B\}$  is a 1-simplex in  $N$  whenever  $|A \cap B| = r - 1$ , for it is clear that one can obtain any  $r$ -subset of  $\underline{n}$  from any other  $r$ -subset in steps that only change one element at a time. So let  $A = \{a_1, \dots, a_{r-1}, a_r\}$  and  $B = \{a_1, \dots, a_{r-1}, a'_r\}$ . Clearly  $|A \cup B| = r + 1$ .

Case  $q \geq 1$ .

$$\begin{aligned} |\underline{n} \setminus (A \cup B)| &= n - |A \cup B| \\ &= n - r - 1 \\ &\geq r - q - 1, \text{ since } n + q \geq 2r. \end{aligned}$$

Therefore there exist  $b_1, \dots, b_{r-q-1} \notin A \cup B$ . Let  $C = \{a_r, a'_r, a_1, \dots, a_{q-1}, b_1, \dots, b_{r-q-1}\}$ .

Thus  $|C \cap A| = |C \cap B| = q$ . So,  $(A, C)$  and  $(B, C) \in E$  and  $\{A, B\} \subseteq \sigma(C)$  follows.

Therefore  $AB$  is a 1-simplex in  $N$ .

Case  $q = 0$ .

$$\begin{aligned} |\underline{n} \setminus (A \cup B)| &= n - |A \cup B| \\ &= n - (r + 1) \\ &= (n - r) - 1 \\ &\geq (r + 1) - 1, \text{ since } n \geq 2r + 1 \\ &\geq r. \end{aligned}$$

Thus there exist  $b_1, \dots, b_r \notin A \cup B$ . Let  $C = \{b_1, \dots, b_r\}$ . Notice that  $|A \cap C| = |B \cap C| = 0$ , so  $(A, C), (B, C) \in E$ . It follows that  $\{A, B\} \subseteq \sigma(C)$  and thus,  $AB$  is



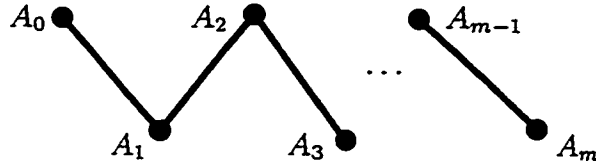
a 1-simplex of  $N$ . □

**Remark 5.2.2** 1. In the case  $q \geq 1$  of the above proof, when  $r = q + 1$ , one has simply  $C = \{a_r, a'_r, a_1, \dots, a_{q-1}\}$ .

2. The main idea in the above proof was to use 1-simplexes  $\{AB\}$  in the neighbourhood simplicial complex where  $|A \cap B| = r - 1$ . This idea leads to the first definition in the following section.

### 5.3 Simple connectivity of the Kneser neighbourhood complexes

**Definition 5.3.1** Let  $A_0, A_1, \dots, A_m$  be a sequence of vertices in a neighbourhood complex  $N(n, r, q)$  such that there is an edge between  $A_i$  and  $A_{i+1}$ .



We call this path a **slow path** if  $|A_i \cap A_{i+1}| = r - 1$ , for  $0 \leq i \leq m - 1$ .

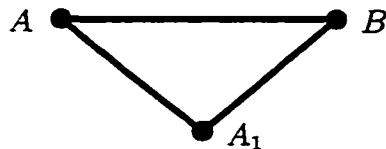
**Examples 5.3.2** For  $N(10, 3, 1)$ , the path  $ABC$  is slow when  $A = \{1, 2, 5\}$ ,  $B = \{1, 2, 7\}$  and  $C = \{2, 3, 7\}$ . The path  $ACB$  is not slow.

**Lemma 5.3.3** Any path in  $N(n, r, q)$  is homotopic to a slow path (with the endpoints of the path always fixed), for any generalized Kneser graph.

**Proof:** Let  $G(n, r, q) = (V, E)$  and  $N = N(n, r, q) = (V, \Sigma)$ . We can restrict ourselves to the case where  $r \geq 2$  since if  $r = 1$ , any path is a slow path.

It clearly suffices to show that any edge in  $N$  is homotopic to a slow path, so let  $A, B \in V$  such that  $AB$  is an edge in  $N$ , i.e.  $\{A, B\} \in \Sigma$ . If  $|A \cap B| = r - 1$  then we are done, since by definition  $AB$  is then a slow edge (path) in  $N$ . Thus we will assume that  $|A \cap B| \leq r - 2$ . We now need to show that  $AB \simeq AA_1A_2 \cdots A_mB$  where  $AA_1A_2 \cdots A_mB$  is a slow path in  $N$  for some  $m \geq 1$ .

Since  $|\underline{n}| = n$  and  $|V| < \infty$ , it suffices to prove that for any edge  $AB$  with  $|A \cap B| \leq r - 2$ , there exists  $A_1 \in V$  such that  $|A \cap A_1| = r - 1 > |A \cap B|$  and  $|A_1 \cap B| > |A \cap B|$ .



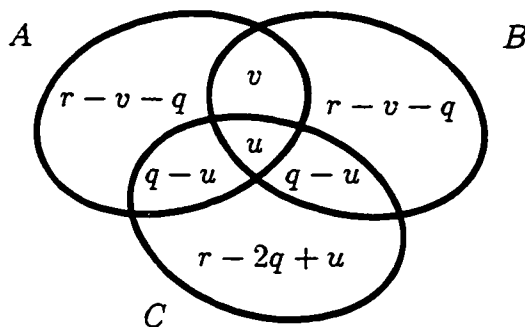
For the edge  $AA_1$  is now slow, and if  $|A_1 \cap B| \leq r - 2$ , then we can repeat this procedure with the edge  $A_1B$  to find another vertex  $A_2$  such that  $|A_1 \cap A_2| = r - 1$  and  $|A_2 \cap B| > |A_1 \cap B|$ , until we have  $|A_m \cap B| = r - 1$  for some  $m \in \mathbb{N}$ .

Keep in mind that from the above inequalities  $A, A_1, A_2, \dots, A_m, B$  are all distinct.

Let  $A = \{a_1, \dots, a_s, b_1, \dots, b_t\}$ ,  $B = \{a_1, \dots, a_s, c_1, \dots, c_t\}$ , where  $s + t = r$  and  $s = |A \cap B| \leq r - 2$  (thus  $t \geq 2$ ).

Since  $\{A, B\} \in \Sigma$ , this implies that there exists a vertex  $C \in V$  such that  $(A, C), (B, C) \in E$ , i.e.  $|A \cap C| = q = |B \cap C|$ . Thus  $\{A, B\} \subseteq \sigma(C)$ .

Now let  $u = |A \cap B \cap C|$  and  $v = s - u$ , i.e.  $u + v = s$ . The following Venn diagram illustrates the situation.



Therefore

$$\begin{aligned} |(A \cap C) \setminus B| &= q - u = |(B \cap C) \setminus A| \\ |A \setminus (B \cup C)| &= r - v - q = |B \setminus (A \cup C)|. \end{aligned}$$

Now we have two different cases to be concerned with, one where  $r - v - q > 0$  and the other when  $r - v - q = 0$ .

Let us consider the first case when  $r - v - q > 0$ , i.e.  $A \setminus (B \cup C), B \setminus (A \cup C) \neq \emptyset$ .

Therefore there exists  $a \in A \setminus (B \cup C)$  and  $b \in B \setminus (A \cup C)$ . Let  $A_1 = (A \setminus \{a\}) \cup \{b\}$ .

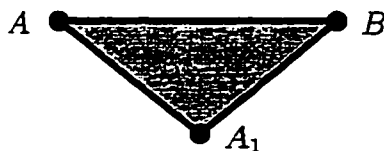
Then  $A_1 \cap C = A \cap C$  since  $a, b \notin C$ . We now have

$$|A_1 \cap A| = r - 1 > r - 2 \geq r - t = s,$$

$$|A_1 \cap B| = s + 1 > s, \text{ and}$$

$$|C \cap A_1| = q \text{ since } A \cap C = A_1 \cap C.$$

Hence  $A_1 \in \sigma(C)$ , since  $|C \cap A_1| = q$ . It follows that  $\{A, A_1, B\} \subseteq \sigma(C)$ .



Therefore the edge  $AB$  is homotopic to the path  $AA_1B$ . Thus the lemma is proved for this case.

Now let us consider the case where  $r - v - q = 0$ , i.e.  $A \setminus (B \cup C), B \setminus (A \cup C) = \emptyset$ . Therefore  $r - v = q$  and  $q - u = r - v - u = r - v - (s - v) = r - s = t \geq 2$ . Therefore there exists  $a \in (A \cap C) \setminus B$  and similarly there exists  $b \in (B \cap C) \setminus A$ , i.e.  $a \notin B$  and  $b \notin A$ . Also note that  $a \neq b$ . Let  $A_1 = (A \setminus \{a\}) \cup \{b\}$ . We now have

$$\begin{aligned}
|C \cap A_1| &= |C \cap [(A \setminus \{a\}) \cup \{b\}]| \\
&= |[C \cap (A \setminus \{a\})] \cup (C \cap \{b\})| \\
&= |C \cap (A \setminus \{a\})| + |C \cap \{b\}| - |C \cap (A \setminus \{a\}) \cap \{b\}| \\
&= (q - 1) + |C \cap \{b\}| - |C \cap (A \setminus \{a\}) \cap \{b\}|, \text{ since } a \in C \\
&= (q - 1) + 1 - |C \cap (A \setminus \{a\}) \cap \{b\}|, \text{ since } b \in C, \\
&= (q - 1) + 1 - 0, \text{ since } b \notin A, \\
&= q.
\end{aligned}$$

Therefore  $(A_1, C) \in E$ . It follows, once again that  $\{A, A_1, B\} \subseteq \sigma(C)$ , and hence the edge  $AB$  is homotopic to the path  $AA_1B$ .  $\square$

**Corollary 5.3.4** *Any loop in  $N(n, r, q)$  is homotopic to a slow loop (with the base-point of the loop always fixed), for any generalized Kneser graph.*

Proof: Let  $N = N(n, r, q) = (V, \Sigma)$ . Suppose that  $\lambda = A_0 A_1 \cdots A_n A_0$  is a loop in  $N$ , where  $A_i \in V$  for all  $0 \leq i \leq n$ . We know that by the definition, a loop is a path which contains the same initial and terminal points. Therefore by Lemma 5.3.3,  $\lambda \simeq \lambda'$  where  $\lambda'$  is a slow path with the endpoints fixed. Therefore  $\lambda'$  is a slow loop, since  $A_0$  is both the initial and terminal point.  $\square$

Recall that  $G(n, r, r)$  is a graph with no edges,  $G(n, n, q)$  is a graph with only one vertex and  $G(n, 1, 0)$  is the complete graph on  $n$  vertices. These are all rather trivial cases, thus we only consider  $n > r > q \geq 0$  and  $r > 1$ . In fact these cases will

be ruled out in the remainder of the thesis with the assumption that  $n + q \geq 2r + 2$ ,  $r \geq 2$  and  $r > q$ .

**Definition 5.3.5** For any path  $\mu = A_1 A_2 \dots A_s$  in  $N(n, r, q)$ , the support of  $\mu$  is denoted  $\text{supp}(\mu) = \min\{m : A_i \subseteq \underline{m}, 1 \leq i \leq s\}$ .

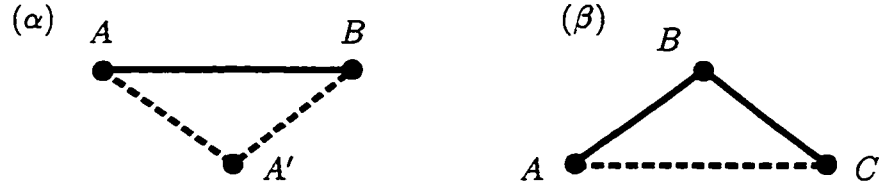
Notice that for  $N(n, r, q)$ ,  $r \leq \text{supp}(\mu) \leq n$ . Our main theorem for simple connectivity is proved next as Theorem 5.3.6. This theorem handles all cases except for  $G(n, 2, 1)$  and  $G(n, 1, 0)$ . Theorem 5.3.7 completes the remaining cases.

**Theorem 5.3.6**  $N(n, r, q)$  is 1-connected when  $n + q \geq 2r + 2$ , and  $r \geq q + 2$  or  $q \geq 2$ .

Proof: Let  $N = N(n, r, q) = (V, \Sigma)$  and  $G = G(n, r, q) = (V, E)$ . By Theorem 5.2.1,  $X_G$  is path connected, so it remains to show that  $\pi_1(X_G) = 0$ , i.e. any loop is homotopic to the trivial loop. Due to the symmetric properties of the Kneser graphs, and using Lemma 5.1.6, we can without loss of generality take the vertex  $A_0 = \{1, 2, \dots, r\} \in V$  as the basepoint of  $X_G$  and suppose that  $\lambda = A_0 A_1 \dots A_s A_0$  is a loop in  $N$ . In fact, by Corollary 5.3.4, we can suppose that  $\lambda$  is a slow loop, since we may replace it by any homotopic loop. Suppose  $\text{supp}(\lambda) = m$ . If  $m = r$  then  $A_i = A_0$ ,  $1 \leq i \leq s$ , whence  $\lambda$  is just the constant loop  $A_0$  and we are done. Our plan is to suppose that  $m \geq r + 1$ , and then show that  $\lambda$  is homotopic to a loop  $\mu$  with  $\text{supp}(\mu) \leq m - 1$ . By applying this technique a finite number of times, we will eventually get  $\lambda \simeq \lambda'$  with  $\text{supp}(\lambda') = r$ . But then, as in the case  $m = r$ ,  $\lambda'$  is just the constant loop  $A_0$ , which will complete the proof.

To carry out this reduction, we will use two procedures, called  $(\alpha)$  and  $(\beta)$ . These are as follows.

- ( $\alpha$ ) Given a slow edge in  $N$ , say  $AB$ , with  $m \in A \cap B$  and  $\text{supp}(AB) = m$ , we shall create a homotopic slow path  $AA'B$  with  $m \notin A'$  (thus any edges with consecutive  $m$ 's are eliminated).



- ( $\beta$ ) Given a slow path  $ABC$  in  $N$  with  $m \notin A, C$  and  $m \in B$  and  $\text{supp}(ABC) = m$ , we show  $ABC \simeq AC$  (and thereby eliminate all remaining  $m$ 's).

- ( $\alpha$ ): From the hypotheses on  $A$  and  $B$  we may write  $A = \{a_1, \dots, a_{r-2}, a_{r-1}, m\}$  and  $B = \{a_1, \dots, a_{r-2}, a'_{r-1}, m\}$ , where  $a_1, \dots, a_{r-1}, a'_{r-1} < m$  and are all distinct. Let  $A' = \{a_1, \dots, a_{r-2}, a_{r-1}, a'_{r-1}\}$ . We need to show that  $\{A, A', B\} \subseteq \sigma(C)$  for some  $C \in V$ . Notice that  $|A \cup B| = r + 1$ . Therefore

$$\begin{aligned}
 n - |A \cup B| &= n - (r + 1) \\
 &= n - r - 1 \\
 &\geq (r + 2 - q) - 1 \\
 &= r - q + 1,
 \end{aligned}$$

i.e. there exist  $b_1, \dots, b_{r-q+1} \notin A \cup B$ . Let  $C = \{a_1, \dots, a_q, b_1, b_2, \dots, b_{r-q}\}$ . Therefore  $|C \cap A| = |C \cap B| = |C \cap A'| = q$ , which implies that  $\{A, B, A'\} \subseteq \sigma(C)$ . As a consequence,  $AB \simeq ABA'$ , and procedure ( $\alpha$ ) is justified.

- ( $\beta$ ): Having carried out ( $\alpha$ ), it remains to consider a slow path  $ABC$  in  $N$  such that  $m \in B$ ,  $m \notin A, C$  and  $\text{supp}(ABC) = m$ . Then  $A = \{a_1, \dots, a_{r-1}, a_r\}$ ,  $B =$

$\{a_1, \dots, a_{r-1}, m\}$  and  $C = \{a_1, \dots, a_{r-1}, a'_r\}$ , with  $a_1, \dots, a_r, a'_r < m$  and all distinct (the case  $a_r = a'_r$  can also occur but is trivial, the path  $ABA$  is homotopic to the constant path  $A$ ).

Clearly  $|A \cup B \cup C| = r + 2$ . Therefore,

$$\begin{aligned}
 |\underline{n} \setminus (A \cup B \cup C)| &= n - |A \cup B \cup C| \\
 &= n - (r + 2) \\
 &\geq (2r - q + 2) - r - 2 \\
 &= r - q \\
 &\geq 2, \text{ since } r \geq q + 2.
 \end{aligned}$$

This implies that there exist  $b_1, \dots, b_{r-q} \notin A \cup B \cup C$ . Thus there exists  $D \in V$  such that  $D = \{a_1, \dots, a_q, b_1, \dots, b_{r-q}\}$ . In fact,  $|D \cap A| = |D \cap B| = |D \cap C| = q$ . Therefore  $\{A, B, C\} \subseteq \sigma(D)$ , and the path  $ABC$  is homotopic to the path  $AC$ , i.e.  $ABC \simeq AC$ .  $\square$

Let us now turn to the two cases not covered by this theorem. The first case is  $N(n, 1, 0)$ , with  $n \geq 4$ . This case is trivial since by Remark 3.3.2  $G(n, 1, 0) = K_n$ , so  $X_{K_n} \cong S^{n-2}$  (cf. 4.3.13), and  $\text{conn } S^{n-2} = n - 3 \geq 1$  (cf. 4.3.3 (3)). It remains to consider the case where  $r = 2$ ,  $q = 1$ , and  $n \geq 2r + 2 - q = 5$ .

**Theorem 5.3.7**  $N(n, 2, 1)$  is 1-connected when  $n \geq 5$ .

Proof: Let  $N = (V, \Sigma) = N(n, r, q), G = (V, E) = G(n, r, q)$ . As in the proof of Theorem 5.3.6,  $X_G$  is path connected, therefore consider (without loss of generality) a loop  $\lambda = A_0 A_1 \cdots A_{s-1} A_0$  in  $X_G$ , with  $A_0 = \{1, 2\}$ . Again, due to Corollary 5.3.4, we can assume  $\lambda$  is a slow loop. Let  $\text{supp}(\lambda) = m$ . As before, we can assume that

$m \geq 3$ , if  $m = 2$  then  $\lambda$  is the constant loop. We would like to show that  $\lambda \simeq \lambda'$  where  $\text{supp}(\lambda') \leq m - 1$ . Thus we consider the same reduction steps as the ones found in the proof of Theorem 5.3.6.

( $\alpha$ ): From the hypothesis on  $A$  and  $B$  we may write  $A = \{a, m\}$  and  $B = \{b, m\}$  for some  $a, b \in \underline{n}$  where  $a \neq b$ . First assume that  $m \geq 4$ . Clearly  $|A \cup B| = 3$ .

$$\begin{aligned}
 |\underline{n} \setminus [(A \cup B) \cup \{m, \dots, n\}]| & \\
 &= n - |(A \cup B) \cup \{m, \dots, n\}| \\
 &= n - |A \cup B| - |\{m, \dots, n\}| + |(A \cup B) \cap \{m, \dots, n\}| \\
 &= n - 3 - (n - m + 1) + 1, \text{ since } m \in A \cup B \\
 &= m - 3 \\
 &\geq 1, \text{ from the assumption on } m.
 \end{aligned}$$

Therefore there exists  $m' \notin A \cup B$  and  $m' < m$ .

Let  $A' = \{a, m'\}$  and  $B' = \{b, m'\}$ . Let  $C = \{a, b\}$ . Clearly,  $|C \cap A| = |C \cap B| = |C \cap A'| = |C \cap B'| = 1$ . Hence,  $\{A, B, A', B'\} \subseteq \sigma(C)$ . Therefore,  $AA'B'B$  is a slow path in  $N$  and  $m \notin A', B'$ . Thus ( $\alpha$ ) is implemented, for  $m \geq 4$ .

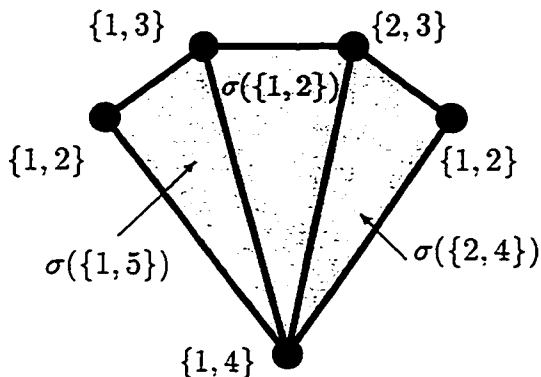
( $\beta$ ): Suppose we have a slow path  $ABC$  in  $N$  such that  $A = \{a, a'\}$ ,  $B = \{a, m\}$ , and  $C = \{a, c\}$ . Once again, it is clear that  $|A \cup B \cup C| = 4$ . Now let us consider the complement of  $A \cup B \cup C$ .

$$\begin{aligned}
 |\underline{n} \setminus (A \cup B \cup C)| &= n - |A \cup B \cup C| \\
 &= n - 4 \\
 &\geq 1, \text{ since } n \geq 5.
 \end{aligned}$$



Therefore, there exists  $d \notin A \cup B \cup C$ . Let  $D = \{a, d\}$ . Clearly  $|D \cap A| = |D \cap B| = |D \cap C| = 1$ . Therefore,  $\{A, B, C\} \subseteq \sigma(D)$ . Thus  $ABC \simeq AC$ , and  $(\beta)$  is implemented.

Applying  $(\alpha)$  and  $(\beta)$  a finite number of times, with  $m \geq 4$ , leaves us with a loop  $\mu \simeq \lambda$  where  $\text{supp}(\mu) \leq 3$ . It follows that  $\mu$  can only contain the three vertices  $A_0 = \{1, 2\}$ ,  $B = \{1, 3\}$  and  $C = \{2, 3\}$ . It is then clear that  $\mu$  is composed of sections such as  $A_0BCA_0$  (or its reverse) and homotopically trivial sections such as  $A_0BA_0$ , etc. However, letting  $D = \{1, 4\}$  the loop  $A_0BCA_0$  is homotopic to the loop  $A_0DA_0$ , as shown by the following diagram, and hence homotopic to the trivial loop. Note the identification of the two  $\{1, 2\}$  vertices in this diagram (and hence also the two edges from  $\{1, 2\}$  to  $\{1, 4\}$ ).



□

## 5.4 Computations of specific examples

**Notation 5.4.1** In order to make this section a little easier to read, we will not display the data in exactly the same format as it appears in the files which are generated by the homology program. Instead of displaying the data “vertically” as they appear in the files, we will display them horizontally. For example if the file

contained the data as follows, i.e. with more than one integer per line

1	2	3
4	5	6
7	8	9
10	11	12

we will display it as

1	2	3	4	5	6	7	8	9	10	11	12
---	---	---	---	---	---	---	---	---	----	----	----

or

1	2	3	4	5	6
7	8	9	10	11	12

(5.1)

If a file contains only one integer per line for example A below, then we will use either *B* or *C*'s format.

A											
1											
2											
3											
4	,	B									
5		<table style="border-collapse: collapse; margin: 0 auto;"> <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr style="border-top: 1px solid black;"><td>5</td><td>6</td><td>7</td><td>8</td></tr> </table>	1	2	3	4	5	6	7	8	
1	2	3	4								
5	6	7	8								
6											
7											
8											
			C								
			<table border="1" style="display: inline-table; margin: 0 auto;"> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> </table>	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8				
			(5.2)								

To show the reader some of the output as it is generated by the program, we have adopted the notation of the program for this section only. Instead of displaying the edges of the Kneser graph as a set of ordered pairs, i.e.  $\{(v_1, v_2), (v_3, v_4)\}$ , they will be displayed as a list of lists, i.e.  $[[v_1, v_2], [v_3, v_4]]$ . Similarly for the neighbourhood

maximal simplexes, instead of displaying them as a power set of sets, we will display them as a list of lists. For example if  $\{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}\}$  are the neighbourhood maximal simplexes of a simplicial complex, then they would be represented as  $[[v_1, v_2, v_3], [v_4, v_5, v_6]]$  in the program. We will clearly state if we are referring to the edges of the Kneser graph or the neighbourhood maximal simplexes.

#### 5.4.1 Computation of the homology of $N(5, 2, 0)$

Here is an explicit example of the homology program.

We will compute the homology groups of the neighbourhood complex of the Petersen graph  $G(5, 2, 0)$ . For an illustration see Figure 3.6 on page 51. One of the first things the computer will do is generate the vertices of the graph  $G(5, 2, 0)$  which are represented below.

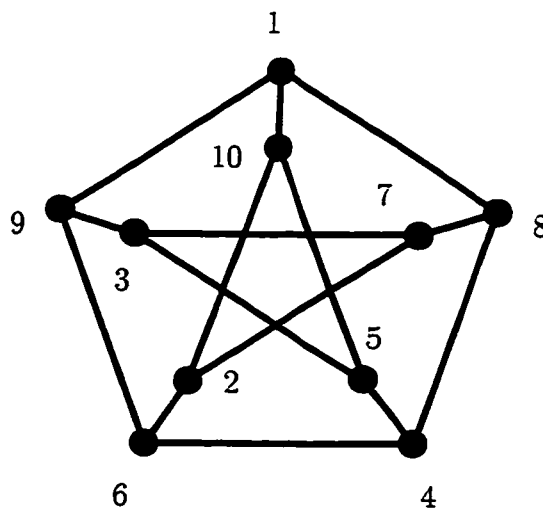
$$\begin{array}{l} \{1, 2\} \quad \{2, 4\} \\ \{1, 3\} \quad \{2, 5\} \\ \{1, 4\} \quad \{3, 4\} \\ \{1, 5\} \quad \{3, 5\} \\ \{2, 3\} \quad \{4, 5\}. \end{array}$$

Then it relabels the vertices and determines the edges in the graph  $G(5, 2, 0)$ .

The list of edges in the graph is contained in the file called kn520.

$$\begin{array}{l} \{1, 2\} \leftrightarrow 1 \quad \{2, 4\} \leftrightarrow 6 \\ \{1, 3\} \leftrightarrow 2 \quad \{2, 5\} \leftrightarrow 7 \\ \{1, 4\} \leftrightarrow 3 \quad \{3, 4\} \leftrightarrow 8 \\ \{1, 5\} \leftrightarrow 4 \quad \{3, 5\} \leftrightarrow 9 \\ \{2, 3\} \leftrightarrow 5 \quad \{4, 5\} \leftrightarrow 10. \end{array}$$

Thus we have the following graph.



The edges in the Kneser graph  $G(5, 2, 0)$  are:  $[[7, 8], [6, 9], [5, 10], [4, 5], [4, 6], [4, 8], [3, 5], [3, 7], [3, 9], [2, 6], [2, 7], [2, 10], [1, 8], [1, 9], [1, 10]]$ . Then  $\sigma(v)$ , the set of neighbours of  $v$ , is computed for each  $v \in V$  (recall that  $\sigma(v)$  was defined in Notation 5.1.1).

$v$	$\sigma(v)$			$v$	$\sigma(v)$		
1	8	9	10	6	2	4	9
2	6	7	10	7	2	3	8
3	5	7	9	8	1	4	7
4	5	6	8	9	1	3	6
5	3	4	10	10	1	2	5

Therefore the neighbourhood maximal simplexes are:  $[[6, 7, 10], [5, 7, 9], [3, 4, 10], [5, 6, 8], [1, 3, 6], [1, 2, 5], [8, 9, 10], [2, 3, 8], [1, 4, 7], [2, 4, 9]]$ . This is printed into the file *kn520* for reference. Figure 5.1 on page 87 shows the relabeling of the neighbourhood simplicial complex in Figure 4.1 on page 65. The program now looks at the corresponding chain complex.

Since all the maximal simplexes have dimension 2, it follows that  $C_i(N(5, 2, 0)) = 0$  for all  $i \geq 3$ . The program proceeds to determine the basis elements of  $C_2$  and in doing so creates the files *allci5202* and *Ci52021*, where both files contain the following data.

1	2	5	8	9	10	2	3	8	1	4	7	6	7	10
5	7	9	2	4	9	3	4	10	5	6	8	1	3	6

(5.3)

The program proceeds, determining the basis of  $C_1$  and the flag matrix associated with it (as shown below). The flag matrix was created to speed up the comparisons between elements in the neighbourhood simplicial complex, which are done when constructing the differential matrices. Using a one-to-one function (the hash function) each basis element of  $C_i(G(n, r, q))$  has its own “hash value” for a fixed  $i$  which represents that basis element. The flag matrix is a two-column matrix. The first column represents the hash value of each basis element of  $C_i(G(n, r, q))$  for a fixed  $i$ . The second column represents the row in which they appear in the *allcinrqi* file/matrix. The elements of the flag matrix are also arranged in increasing order with respect to the first entry of each row.

For example, consider the entry 10, 5 in the flag array. The 10 represents the edge  $\{2, 3\}$  in the neighbourhood complex. It receives the ordinal 10 since it is the 10th entry on the list of all 45 pairs  $\{i, j\}$ , where  $1 \leq i < j \leq 10$ . The second coordinate, 5, denotes the fact that in the first display the basis element  $\{2, 3\}$  is the 5th entry. Further examples are given in Appendix B.1.6 on page 148.

A basis for  $C_1$ .

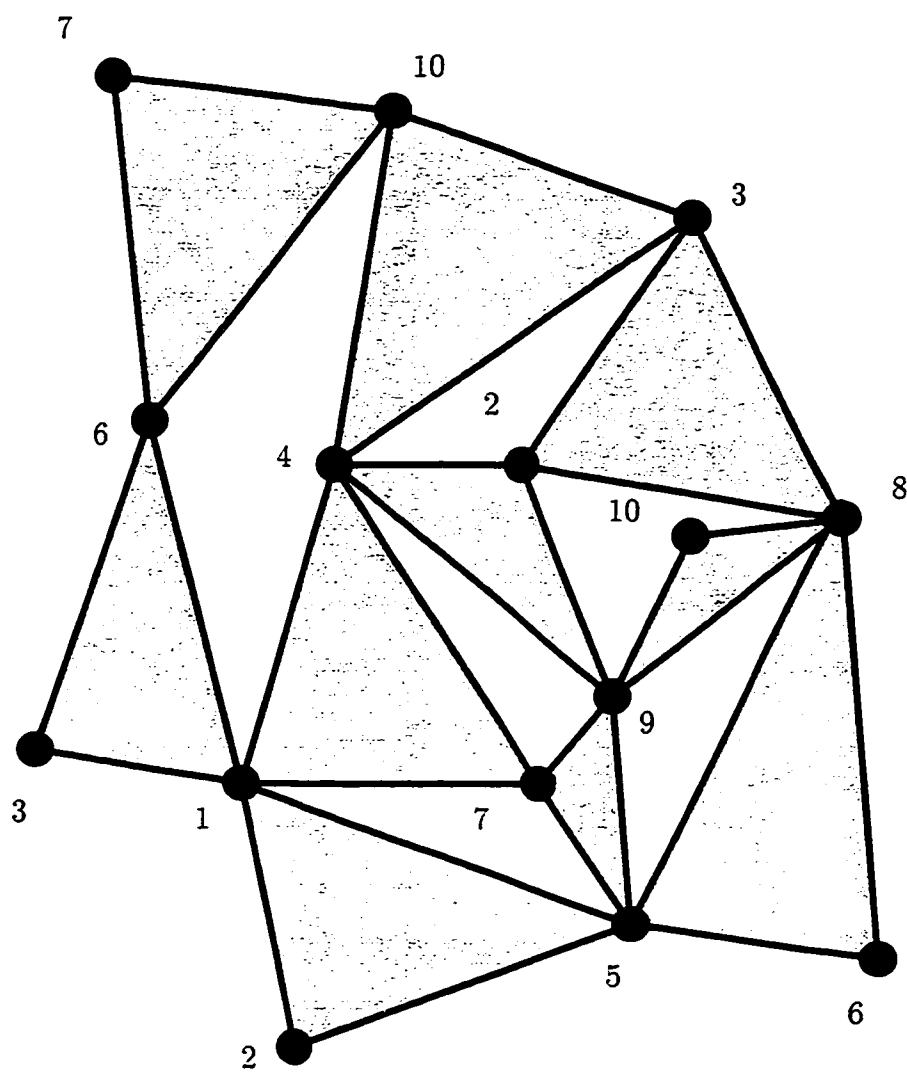
5 7	5 9	8 10	9 10	2 3	2 8	3 8	1 4	2 5	8 9
1 3	1 6	3 6	1 5	5 8	6 8	4 10	5 6	7 9	3 4
3 10	4 9	1 7	4 7	6 7	6 10	1 2	7 10	2 4	2 9

A flag for  $C_1$ .

1 27	2 11	3 8	4 14	5 12	6 23	10 5	11 29	12 9	15 6
16 30	18 20	20 13	22 7	24 21	27 24	29 22	30 17	31 18	32 1
33 15	34 2	36 25	37 16	39 26	41 19	42 28	43 10	44 3	45 4

Then the homology program determines a basis for  $C_i(N(5,2,0))$ . It writes out the number of elements there are in the basis of  $C_i(N(5,2,0))$  and lists all the basis elements of the  $C_i(N(5,2,0))$ , if there are less than 20 elements, in the file called kn520. It also creates two other types files allci520i, which contains the entire list of the basis elements for  $C_i(N(5,2,0))$ , and Ci520iq where  $i$  is the  $i$ th cell, and  $q$  is the number of multiples of twenty elements in  $C_i(N(5,2,0))$  plus any remainders. The first file is for the user's records, if they wish to keep a record. The second file is for the computer, to help with the derivation of the differential matrices. In fact, this forces the computer to use more storage memory and less "active" memory.

Using the flag matrix for  $C_1$ , and the file Ci52021, the program determines the differential matrix associated with  $d_2$  and places the information into the file called differential2. The first two entries in the following display tell us that  $d_2$  is a  $10 \times 30$  matrix. The first 30 entries following 10,30 (i.e. the first row of  $d_2$ ) express the equation  $d_2\{1,2,5\} = \{1,2\} - \{1,5\} + \{2,5\} = (27) - (14) + (9)$ , as translated by

Figure 5.1:  $N(5, 2, 0)$

the flag for  $C_1$  on p. 86. Similarly, the next 30 entries give  $d_2\{8,9,10\}$ , etc.

10	30	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1	1	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	-1	0	1	0	0	1	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	1	-1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0																			

The abelian group  $C_0(N(5,2,0))$  is then calculated and the data below is stored in both allci5200 and Ci52001. The associated flag matrix is also determined.

$C_0$

9	5	4	3	6	7	2	10	8	1
---	---	---	---	---	---	---	----	---	---

flag for  $C_0$

1	10	2	7	3	4	4	3	5	2	6	5	7	6	8	9	9	1	10	8
---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	---



Now the program determines the differential matrix of  $d_1$ , using the files Ci52011 and Ci52012 as well as the flag matrix for  $C_0$ , and writes it in the appropriate form for Arne Storjohann's ISmith program as shown below.

30	10	0	-1	0	0	0	1	0	0	0	0	1	-1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	-1	0	-1	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	-1	0	0	0	0	0
0	0	0	0	-1	0	1	0	0	0	0	-1	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	-1	0	1	0	0	0	0	-1	0
0	0	1	0	0	0	0	0	0	0	-1	0	0	0	0	1	0	0
0	0	0	-1	0	0	0	0	1	0	0	0	0	-1	0	0	0	-1
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	-1	0	-1
0	0	0	0	0	0	1	0	0	0	0	0	-1	0	0	0	1	0
0	0	-1	0	0	0	0	1	0	0	0	-1	0	0	1	0	0	0
0	0	1	0	0	0	0	-1	0	0	0	0	0	0	1	-1	0	0
0	0	0	0	0	0	0	-1	0	0	0	1	0	0	1	0	-1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0
-1	0	0	1	0	0	0	0	0	0	0	0	-1	1	0	0	0	0
0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	1	0
0	-1	0	0	0	0	0	-1	0	1	0	0	0	0	1	0	0	0
-1	0	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	0	0

We have now computed all the chains and differentials in the following sequence.

$$0 \xrightarrow{d_3} C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} 0.$$

We also know that from the above sequence,  $\ker d_0 = C_0$  and  $\text{im } d_0 = 0$ . Using Arne

Storjohann's ISmith program, we can determine the kernel and image of  $d_2$  and  $d_1$  shown below.

$$\begin{aligned} \ker d_2 = 0 \quad \ker d_1 \approx \mathbb{Z}^{21} \quad \ker d_i = 0 \\ \text{im } d_2 \approx \mathbb{Z}^{10} \quad \text{im } d_1 \approx \mathbb{Z}^9 \quad \text{im } d_i = 0, \text{ for all } i \geq 3. \end{aligned}$$

Therefore,

$$\begin{aligned} H_0(N(5, 2, 0)) &= \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^{10} / \mathbb{Z}^9 \approx \mathbb{Z} \\ H_1(N(5, 2, 0)) &= \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{21} / \mathbb{Z}^{10} \approx \mathbb{Z}^{11} \\ H_2(N(5, 2, 0)) &= \ker d_2 / \text{im } d_3 = 0/0 = 0 \\ H_i(N(5, 2, 0)) &= \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 3. \end{aligned}$$

This completes the calculation of the homology of  $N(5, 2, 0)$ . Without going through these details again, we will merely state the homology groups of the given spaces which appear after Section 5.4.2.

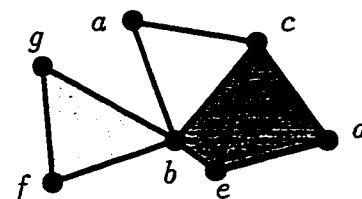
#### 5.4.2 Computation of the Relative Homology of $N(5, 2, 0)$

Before we proceed with the computation of the relative homology of  $N(5, 2, 0)$ , we first give a definition and state two useful theorems.

**Definition 5.4.2** *Let  $K = (V, \Sigma)$  be a simplicial complex (abstract or geometric), and let  $v \in V$  be a vertex of  $K$ . The star of  $v$ , denoted  $\text{star}(v)$ , is the smallest subcomplex of  $K$  containing all simplexes  $\tau \in \Sigma$  for which  $v \in \tau$  (i.e.  $v$  is a vertex of  $\tau$ ).*

**Remark 5.4.3** In many texts this would be called the “closed star” of  $v$ . One may also define the “open star” of  $v$ , cf. [11], p.135, [12], p.114.

**Example 5.4.4** Let  $K = (V, \Sigma)$  be a simplicial complex with  $V = \{a, b, c, d, e, f, g\}$  and maximal simplices  $\{\{a, b\}, \{a, c\}, \{b, f, g\}, \{b, c, d, e\}\}$ . The sketch below illustrates  $K$ .



Then we have the following results:

1.  $star(a) = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}\}$ .
2.  $star(b) = K \setminus \{\{a, c\}\}$ , i.e. the star of  $b$  is the entire simplicial complex with the interior of edge  $\{a, c\}$  removed.
3.  $star(c) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a, c\}, \{b, c\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}, \{d, e\}, \{b, c, d\}, \{b, d, e\}, \{c, d, e\}, \{b, c, e\}, \{b, c, d, e\}\}$ .
4.  $star(d) = \{\{b\}, \{c\}, \{d\}, \{e\}, \{b, c\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}, \{d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}, \{b, c, d, e\}\}$ .
5.  $star(e) = star(d)$ .
6.  $star(f) = \{\{b\}, \{f\}, \{g\}, \{b, f\}, \{b, g\}, \{f, g\}, \{b, f, g\}\}$ .
7.  $star(g) = star(f)$ .

The next theorem is simply stated here, cf. [11], p.153.

**Theorem 5.4.5** For any vertex  $v$  of a simplicial complex  $K$ ,  $|star(v)|$  is contractible.

The final theorem stated here is an elementary consequence of the exact sequence of a pair, and the fact that all homology groups  $H_i(X)$  of a contractible space  $X$  vanish for  $i > 0$ . We omit the proof.

**Theorem 5.4.6** *Let  $(Y, X)$  be a polyhedral pair of spaces with  $X$  contractible. Then, for all  $n > 0$ ,  $H_n(Y) \approx H_n(Y, X)$ .*

Of course, we shall apply these two theorems in the case where  $Y$  is the neighbourhood complex of a Kneser graph, and  $X$  is the star of a vertex of  $Y$ . To show how this can reduce the calculations involved, we now repeat the calculations for  $G(5, 2, 0)$  using the above techniques.

Specifically, in this section we will compute  $H_*(L, M)$  where  $L = N(5, 2, 0)$  and  $M = \text{star}(\{1, 2\})$ . The computation of the homology is the same as in Section 5.4.1 up to and including the neighbourhood maximal simplexes. Then the relative homology program branches off from that point on. The  $M = \text{star}(1)$  is determined to be  $[[1, 4, 7], [1, 3, 6], [1, 2, 5]]$ . We also know, since the maximal simplexes in the neighbourhood complex contain three vertices, that  $C_i(L) = 0$  for all  $i \geq 3$ . Therefore,  $C_i(M) = 0$  for all  $i \geq 3$ . The basis elements of  $C_2(M)$  are as follows:

A basis for  $C_2(M)$

$$\begin{array}{|c|c|c|} \hline 1 & 4 & 7 \\ \hline 1 & 3 & 6 \\ \hline 1 & 2 & 5 \\ \hline \end{array}, \text{ and}$$

a flag for  $C_2(M)$

$$\begin{array}{|c|c|c|c|} \hline 3 & 3 & 11 & 2 \\ \hline 18 & 1 & & \\ \hline \end{array}$$

After the flag matrix for  $C_2(M)$  has been determined, the basis elements of  $C_2(L, M)$  can now be determined, which are as follows.

3	4	10	5	6	8	2	4	9	2	3	8
8	9	10	6	7	10	5	7	9			

Now the program will compute  $C_1(M)$ , its flag matrix and using both of them, it will determine the basis elements of  $C_1(L, M)$  and the flag matrix associated with it. These are shown below.

A basis for  $C_1(M)$

1	2	2	5	1	6	3	6	1	4	1	7	1	3	1	5	4	7
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

a flag matrix for  $C_1(M)$

1	1	2	7	3	5	4	8	5	3	6	6	12	2	20	4	27	9
---	---	---	---	---	---	---	---	---	---	---	---	----	---	----	---	----	---

a basis for  $C_1(L, M)$

4	10	3	10	3	4	7	10	2	3	2	8	3	8
8	9	8	10	9	10	6	7	6	10	5	8	6	8
2	4	2	9	4	9	5	6	5	9	5	7	7	9

, and

a flag for  $C_1(L, M)$

10	5	11	15	15	6	16	16	18	3	22	7	24	2
29	17	30	1	31	18	32	20	33	13	34	19	36	11
37	14	39	12	41	21	42	4	43	8	44	9	45	10

Then the differential matrix,  $D_2$  is determined to be the following

7	21	1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	-1	1	0	0	0	0	0	0	0	0	1	-1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	1	1													

After  $D_2$  has been determined, we have one more differential matrix for the computer to compute, which is  $D_1$ . In order to do this the program will first have to determine  $C_0(M)$ , its flag matrix and then  $C_0(L, M)$ . These are

A basis for  $C_0(M)$

6	3	1	2	7	5	4
---	---	---	---	---	---	---

a flag matrix for  $C_0(M)$ ,

1	3	2	4	3	2	4	7	5	6	6	1	7	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---

a basis for  $C_0(L, M)$

10	9	8
----	---	---

flag for  $C_0(L, M)$

8	3	9	2	10	1
---	---	---	---	----	---

Then the differential matrix,  $D_1$  is determined as follows.

$$\begin{array}{cccccccccccccccc}
 21 & 3 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 \hline
 0 & 1 & 0 & 0 & 1 & 0 & 1 & -1 & 1 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 1 \\
 \hline
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
 \hline
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & & & & & & & 
 \end{array}$$

Piecing all of this together, we have the following chain complex:

$$0 \xrightarrow{d_3} C_2(L, M) \xrightarrow{d_2} C_1(L, M) \xrightarrow{d_1} C_0(L, M) \xrightarrow{d_0} 0.$$

Using Arne Storjohann's ISmith program, we find that

$$\left. \begin{array}{lll}
 \ker d_2 = 0 & \ker d_1 \approx \mathbb{Z}^{18} & \ker d_i = 0 \\
 \operatorname{im} d_2 \approx \mathbb{Z}^7 & \operatorname{im} d_1 \approx \mathbb{Z}^3 & \operatorname{im} d_i = 0
 \end{array} \right\} \text{ for all } i \geq 3.$$

Therefore,

$$\begin{aligned}
 H_0(L, M) &= \ker d_0 / \operatorname{im} d_1 \approx \mathbb{Z}^3 / \mathbb{Z}^3 = 0 \\
 H_1(L, M) &= \ker d_1 / \operatorname{im} d_2 \approx \mathbb{Z}^{18} / \mathbb{Z}^7 \approx \mathbb{Z}^{11} \\
 H_2(L, M) &= \ker d_2 / \operatorname{im} d_3 = 0 / 0 = 0 \\
 H_i(L, M) &= \ker d_i / \operatorname{im} d_{i+1} = 0 / 0 = 0, \text{ for all } i \geq 3.
 \end{aligned}$$

These results agree with those of Section 5.4.1, but the work involved is considerably reduced. This shows the advantage of using relative homology, and is important since the size of the calculations involved grows rapidly with  $n$  (and quickly reaches the limits of powerful computers).

## 5.5 Summary of Results

$$N = \mathbf{N}(4, 1, 0) \approx \mathbf{N}(4, 3, 2)$$

$$\ker d_0 \approx \mathbb{Z}^4 \quad \text{im } d_0 = 0 \quad H_0(N) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^4 / \mathbb{Z}^3 \approx \mathbb{Z}$$

$$\ker d_1 \approx \mathbb{Z}^3 \quad \text{im } d_1 \approx \mathbb{Z}^3 \quad H_1(N) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^3 / \mathbb{Z}^3 = 0$$

$$\ker d_2 \approx \mathbb{Z} \quad \text{im } d_2 \approx \mathbb{Z}^3 \quad H_2(N) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z} / 0 = \mathbb{Z}$$

$$\ker d_i = 0 \quad \text{im } d_i = 0 \quad H_i(N) = \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 3$$

$N = \mathbf{N}(4, 2, 0)$ , which consists of six distinct points.

$$\ker d_0 \approx \mathbb{Z}^6 \quad \text{im } d_0 = 0 \quad H_0(N) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^6 / 0 = \mathbb{Z}^6$$

$$\ker d_i = 0 \quad \text{im } d_i = 0 \quad H_i(N) = \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 1$$

$$N = \mathbf{N}(4, 2, 1)$$

$$\ker d_0 \approx \mathbb{Z}^6 \quad \text{im } d_0 = 0 \quad H_0(N) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^6 / \mathbb{Z}^5 \approx \mathbb{Z}$$

$$\ker d_1 \approx \mathbb{Z}^{10} \quad \text{im } d_1 \approx \mathbb{Z}^5 \quad H_1(N) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{10} / \mathbb{Z}^9 \approx \mathbb{Z}$$

$$\ker d_2 \approx \mathbb{Z}^3 \quad \text{im } d_2 \approx \mathbb{Z}^9 \quad H_2(N) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^3 / \mathbb{Z}^3 = 0$$

$$\ker d_3 = 0 \quad \text{im } d_3 \approx \mathbb{Z}^3 \quad H_3(N) = \ker d_3 / \text{im } d_4 = 0/0 = 0$$

$$\ker d_i = 0 \quad \text{im } d_i = 0 \quad H_i(N) = \ker d_i / \text{im } d_{i+1} = 0/0 = 0 \text{ for all } i \geq 4$$



$$N = \mathbf{N}(5, 1, 0) \approx \mathbf{N}(5, 4, 3)$$

$$\begin{aligned} \ker d_0 &\approx \mathbb{Z}^5 & \text{im } d_0 &= 0 & H_0(N) &= \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^5 / \mathbb{Z}^4 \approx \mathbb{Z} \\ \ker d_1 &\approx \mathbb{Z}^6 & \text{im } d_1 &\approx \mathbb{Z}^4 & H_1(N) &= \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^6 / \mathbb{Z}^6 = 0 \\ \ker d_2 &\approx \mathbb{Z}^4 & \text{im } d_2 &\approx \mathbb{Z}^6 & H_2(N) &= \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^4 / \mathbb{Z}^4 = 0 \\ \ker d_3 &\approx \mathbb{Z} & \text{im } d_3 &\approx \mathbb{Z}^4 & H_3(N) &= \ker d_3 / \text{im } d_4 \approx \mathbb{Z} / 0 \approx \mathbb{Z} \\ \ker d_i &= 0 & \text{im } d_i &= 0 & H_i(N) &= \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 4 \end{aligned}$$

$$N = \mathbf{N}(5, 2, 1) \approx \mathbf{N}(5, 3, 2)$$

$$\begin{aligned} \ker d_0 &\approx \mathbb{Z}^{10} & \text{im } d_0 &= 0 & H_0(N) &= \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^{10} / \mathbb{Z}^9 \approx \mathbb{Z} \\ \ker d_1 &\approx \mathbb{Z}^{36} & \text{im } d_1 &\approx \mathbb{Z}^9 & H_1(N) &= \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{36} / \mathbb{Z}^{36} = 0 \\ \ker d_2 &\approx \mathbb{Z}^{74} & \text{im } d_2 &\approx \mathbb{Z}^{36} & H_2(N) &= \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{74} / \mathbb{Z}^{74} = 0 \\ \ker d_3 &\approx \mathbb{Z}^{61} & \text{im } d_3 &\approx \mathbb{Z}^{74} & H_3(N) &= \ker d_3 / \text{im } d_4 \approx \mathbb{Z}^{61} / \mathbb{Z}^{50} \approx \mathbb{Z}^{11} \\ \ker d_4 &\approx \mathbb{Z}^{10} & \text{im } d_4 &\approx \mathbb{Z}^{50} & H_4(N) &= \ker d_4 / \text{im } d_5 \approx \mathbb{Z}^{10} / \mathbb{Z}^{10} = 0 \\ \ker d_5 &= 0 & \text{im } d_5 &\approx \mathbb{Z}^{10} & H_5(N) &= \ker d_5 / \text{im } d_6 = 0/0 = 0 \\ \ker d_i &= 0 & \text{im } d_i &= 0 & H_i(N) &= \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 6 \end{aligned}$$

$$N = \mathbf{N}(6, 1, 0) \approx \mathbf{N}(6, 5, 4)$$

$$\begin{aligned} \ker d_0 &\approx \mathbb{Z}^6 & \text{im } d_0 &= 0 & H_0(N) &= \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^6 / \mathbb{Z}^5 \approx \mathbb{Z} \\ \ker d_1 &\approx \mathbb{Z}^{10} & \text{im } d_1 &\approx \mathbb{Z}^5 & H_1(N) &= \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{10} / \mathbb{Z}^{10} = 0 \\ \ker d_2 &\approx \mathbb{Z}^{10} & \text{im } d_2 &\approx \mathbb{Z}^{10} & H_2(N) &= \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{10} / \mathbb{Z}^{10} = 0 \\ \ker d_3 &\approx \mathbb{Z}^5 & \text{im } d_3 &\approx \mathbb{Z}^{10} & H_3(N) &= \ker d_3 / \text{im } d_4 \approx \mathbb{Z}^5 / \mathbb{Z}^5 = 0 \\ \ker d_4 &\approx \mathbb{Z} & \text{im } d_4 &\approx \mathbb{Z}^5 & H_4(N) &= \ker d_4 / \text{im } d_5 \approx \mathbb{Z} / 0 \approx \mathbb{Z} \\ \ker d_i &= 0 & \text{im } d_i &= 0 & H_i(N) &= \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 5 \end{aligned}$$

$$N = \mathbf{N}(6, 2, 0) \approx \mathbf{N}(6, 4, 2)$$

$$\begin{aligned} \ker d_0 &\approx \mathbb{Z}^{15} & \operatorname{im} d_0 &= 0 & H_0(N) &= \ker d_0 / \operatorname{im} d_1 \approx \mathbb{Z}^{15} / \mathbb{Z}^{14} \approx \mathbb{Z} \\ \ker d_1 &\approx \mathbb{Z}^{91} & \operatorname{im} d_1 &\approx \mathbb{Z}^{14} & H_1(N) &= \ker d_1 / \operatorname{im} d_2 \approx \mathbb{Z}^{91} / \mathbb{Z}^{91} = 0 \\ \ker d_2 &\approx \mathbb{Z}^{169} & \operatorname{im} d_2 &\approx \mathbb{Z}^{91} & H_2(N) &= \ker d_2 / \operatorname{im} d_3 \approx \mathbb{Z}^{169} / \mathbb{Z}^{150} \approx \mathbb{Z}^{19} \\ \ker d_3 &\approx \mathbb{Z}^{75} & \operatorname{im} d_3 &\approx \mathbb{Z}^{150} & H_3(N) &= \ker d_3 / \operatorname{im} d_4 \approx \mathbb{Z}^{75} / \mathbb{Z}^{75} = 0 \\ \ker d_4 &\approx \mathbb{Z}^{15} & \operatorname{im} d_4 &\approx \mathbb{Z}^{75} & H_4(N) &= \ker d_4 / \operatorname{im} d_5 \approx \mathbb{Z}^{15} / \mathbb{Z}^{15} = 0 \\ \ker d_5 &= 0 & \operatorname{im} d_5 &\approx \mathbb{Z}^{15} & H_5(N) &= \ker d_5 / \operatorname{im} d_6 = 0/0 = 0 \\ \ker d_i &= 0 & \operatorname{im} d_i &= 0 & H_i(N) &= \ker d_i / \operatorname{im} d_{i+1} = 0/0 = 0, \text{ for all } i \geq 6 \end{aligned}$$

$$N = \mathbf{N}(6, 2, 1) = \mathbf{N}(6, 4, 3)$$

$$\begin{aligned} \ker d_0 &\approx \mathbb{Z}^{15} & \operatorname{im} d_0 &= 0 & H_0(N) &= \ker d_0 / \operatorname{im} d_1 \approx \mathbb{Z}^{15} / \mathbb{Z}^{14} \approx \mathbb{Z} \\ \ker d_1 &\approx \mathbb{Z}^{91} & \operatorname{im} d_1 &\approx \mathbb{Z}^{14} & H_1(N) &= \ker d_1 / \operatorname{im} d_2 \approx \mathbb{Z}^{91} / \mathbb{Z}^{91} = 0 \\ \ker d_2 &\approx \mathbb{Z}^{329} & \operatorname{im} d_2 &\approx \mathbb{Z}^{91} & H_2(N) &= \ker d_2 / \operatorname{im} d_3 \approx \mathbb{Z}^{329} / \mathbb{Z}^{329} = 0 \\ \ker d_3 &\approx \mathbb{Z}^{616} & \operatorname{im} d_3 &\approx \mathbb{Z}^{329} & H_3(N) &= \ker d_3 / \operatorname{im} d_4 \approx \mathbb{Z}^{616} / \mathbb{Z}^{525} \approx \mathbb{Z}^{91} \\ \ker d_4 &\approx \mathbb{Z}^{315} & \operatorname{im} d_4 &\approx \mathbb{Z}^{525} & H_4(N) &= \ker d_4 / \operatorname{im} d_5 \approx \mathbb{Z}^{315} / \mathbb{Z}^{315} = 0 \\ \ker d_5 &\approx \mathbb{Z}^{105} & \operatorname{im} d_5 &\approx \mathbb{Z}^{315} & H_5(N) &= \ker d_5 / \operatorname{im} d_6 \approx \mathbb{Z}^{105} / \mathbb{Z}^{105} = 0 \\ \ker d_6 &\approx \mathbb{Z}^{15} & \operatorname{im} d_6 &\approx \mathbb{Z}^{105} & H_6(N) &= \ker d_6 / \operatorname{im} d_7 \approx \mathbb{Z}^{15} / \mathbb{Z}^{15} = 0 \\ \ker d_7 &= 0 & \operatorname{im} d_7 &\approx \mathbb{Z}^{15} & H_7(N) &= \ker d_7 / \operatorname{im} d_8 = 0/0 = 0 \\ \ker d_i &= 0 & \operatorname{im} d_i &= 0 & H_i(N) &= \ker d_i / \operatorname{im} d_{i+1} = 0/0 = 0, \text{ for all } i \geq 8 \end{aligned}$$

$\mathbf{N}(6, 3, 0)$ , which consists of twenty distinct points.

$$\begin{aligned} \ker d_0 &\approx \mathbb{Z}^{20} & \operatorname{im} d_0 &= 0 & H_0(N) &= \ker d_0 / \operatorname{im} d_1 \approx \mathbb{Z}^{20} / 0 \approx \mathbb{Z}^{20} \\ \ker d_i &= 0 & \operatorname{im} d_i &= 0 & H_i(N) &= \ker d_i / \operatorname{im} d_{i+1} = 0/0 = 0, \text{ for all } i \geq 1 \end{aligned}$$

$N = N(6, 3, 1) \approx N(6, 3, 2)$ , is discussed in the next section.

Let  $M = N(6, 3, 1)$  and  $L = \text{star}(\{1, 2, 3\})$ .

$$\begin{array}{lll}
\ker d_0 \approx \mathbb{Z} & \text{im } d_0 = 0 & H_0(L, M) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z} / \mathbb{Z} = 0 \\
\ker d_1 \approx \mathbb{Z}^{17} & \text{im } d_1 \approx \mathbb{Z} & H_1(L, M) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{17} / \mathbb{Z}^{17} = 0 \\
\ker d_2 \approx \mathbb{Z}^{331} & \text{im } d_2 \approx \mathbb{Z}^{17} & H_2(L, M) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{331} / (\mathbb{Z}^{330} \oplus 2\mathbb{Z}) \approx \mathbb{Z}_2 \\
\ker d_3 \approx \mathbb{Z}^{911} & \text{im } d_3 \approx \mathbb{Z}^{330} \oplus 2\mathbb{Z} & H_3(L, M) = \ker d_3 / \text{im } d_4 \approx \mathbb{Z}^{911} / \mathbb{Z}^{770} \approx \mathbb{Z}^{141} \\
\ker d_4 \approx \mathbb{Z}^{616} & \text{im } d_4 \approx \mathbb{Z}^{770} & H_4(L, M) = \ker d_4 / \text{im } d_5 \approx \mathbb{Z}^{616} / \mathbb{Z}^{616} = 0 \\
\ker d_5 \approx \mathbb{Z}^{308} & \text{im } d_5 \approx \mathbb{Z}^{616} & H_5(L, M) = \ker d_5 / \text{im } d_6 \approx \mathbb{Z}^{308} / \mathbb{Z}^{308} = 0 \\
\ker d_6 \approx \mathbb{Z}^{88} & \text{im } d_6 \approx \mathbb{Z}^{308} & H_6(L, M) = \ker d_6 / \text{im } d_7 \approx \mathbb{Z}^{88} / \mathbb{Z}^{88} = 0 \\
\ker d_7 \approx \mathbb{Z}^{11} & \text{im } d_7 \approx \mathbb{Z}^{88} & H_7(L, M) = \ker d_7 / \text{im } d_8 \approx \mathbb{Z}^{11} / \mathbb{Z}^{11} = 0 \\
\ker d_8 = 0 & \text{im } d_8 \approx \mathbb{Z}^{11} & H_8(L, M) = \ker d_8 / \text{im } d_9 = 0 / 0 = 0 \\
\ker d_i = 0 & \text{im } d_i = 0 & H_i(L, M) = \ker d_i / \text{im } d_{i+1} = 0 / 0 = 0, \text{ for all } i \geq 9
\end{array}$$

$N = N(7, 1, 0) \approx N(7, 6, 5)$

$$\begin{array}{lll}
\ker d_0 \approx \mathbb{Z}^7 & \text{im } d_0 = 0 & H_0(N) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^7 / \mathbb{Z}^6 \approx \mathbb{Z} \\
\ker d_1 \approx \mathbb{Z}^{15} & \text{im } d_1 \approx \mathbb{Z}^6 & H_1(N) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{15} / \mathbb{Z}^{15} = 0 \\
\ker d_2 \approx \mathbb{Z}^{20} & \text{im } d_2 \approx \mathbb{Z}^{15} & H_2(N) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{20} / \mathbb{Z}^{20} = 0 \\
\ker d_3 \approx \mathbb{Z}^{15} & \text{im } d_3 \approx \mathbb{Z}^{20} & H_3(N) = \ker d_3 / \text{im } d_4 \approx \mathbb{Z}^{15} / \mathbb{Z}^{15} = 0 \\
\ker d_4 \approx \mathbb{Z}^6 & \text{im } d_4 \approx \mathbb{Z}^{15} & H_4(N) = \ker d_4 / \text{im } d_5 \approx \mathbb{Z}^6 / \mathbb{Z}^6 = 0 \\
\ker d_5 \approx \mathbb{Z} & \text{im } d_5 \approx \mathbb{Z}^6 & H_5(N) = \ker d_5 / \text{im } d_6 \approx \mathbb{Z} / 0 \approx \mathbb{Z} \\
\ker d_i = 0 & \text{im } d_i = 0 & H_i(N) = \ker d_i / \text{im } d_{i+1} = 0 / 0 = 0, \text{ for all } i \geq 6
\end{array}$$

$$N = \mathbf{N}(7, 2, 0) \approx \mathbf{N}(7, 5, 3)$$

$$\begin{array}{lll}
\ker d_0 \approx \mathbb{Z}^{21} & \text{im } d_0 = 0 & H_0(N) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^{21} / \mathbb{Z}^{20} \approx \mathbb{Z} \\
\ker d_1 \approx \mathbb{Z}^{190} & \text{im } d_1 \approx \mathbb{Z}^{20} & H_1(N) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{190} / \mathbb{Z}^{190} = 0 \\
\ker d_2 \approx \mathbb{Z}^{1035} & \text{im } d_2 \approx \mathbb{Z}^{190} & H_2(N) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{1035} / \mathbb{Z}^{1035} = 0 \\
\ker d_3 \approx \mathbb{Z}^{2335} & \text{im } d_3 \approx \mathbb{Z}^{1035} & H_3(N) = \ker d_3 / \text{im } d_4 \approx \mathbb{Z}^{2335} / \mathbb{Z}^{2296} \approx \mathbb{Z}^{39} \\
\ker d_4 \approx \mathbb{Z}^{2576} & \text{im } d_4 \approx \mathbb{Z}^{2296} & H_4(N) = \ker d_4 / \text{im } d_5 \approx \mathbb{Z}^{2576} / \mathbb{Z}^{2576} = 0 \\
\ker d_5 \approx \mathbb{Z}^{1764} & \text{im } d_5 \approx \mathbb{Z}^{2576} & H_5(N) = \ker d_5 / \text{im } d_6 \approx \mathbb{Z}^{1764} / \mathbb{Z}^{1764} = 0 \\
\ker d_6 \approx \mathbb{Z}^{756} & \text{im } d_6 \approx \mathbb{Z}^{1764} & H_6(N) = \ker d_6 / \text{im } d_7 \approx \mathbb{Z}^{756} / \mathbb{Z}^{756} = 0 \\
\ker d_7 \approx \mathbb{Z}^{189} & \text{im } d_7 \approx \mathbb{Z}^{756} & H_7(N) = \ker d_7 / \text{im } d_8 \approx \mathbb{Z}^{189} / \mathbb{Z}^{189} = 0 \\
\ker d_8 \approx \mathbb{Z}^{21} & \text{im } d_8 \approx \mathbb{Z}^{189} & H_8(N) = \ker d_8 / \text{im } d_9 \approx \mathbb{Z}^{21} / \mathbb{Z}^{21} = 0 \\
\ker d_9 = 0 & \text{im } d_9 \approx \mathbb{Z}^{21} & H_9(N) = \ker d_9 / \text{im } d_{10} = 0/0 = 0 \\
\ker d_i = 0 & \text{im } d_i = 0 & H_i(N) = \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 10
\end{array}$$

$$N = \mathbf{N}(7, 2, 1) \approx \mathbf{N}(7, 5, 4)$$

Let  $L = N(7, 2, 1)$  and  $M = \text{star}(\{1, 2\})$ .

$$\begin{array}{lll} \ker d_0 = 0 & \text{im } d_0 = 0 & H_0(L, M) = \ker d_0 / \text{im } d_1 = 0/0 = 0 \\ \ker d_1 \approx \mathbb{Z}^{20} & \text{im } d_1 = 0 & H_1(L, M) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{20} / \mathbb{Z}^{20} = 0 \\ \ker d_2 \approx \mathbb{Z}^{280} & \text{im } d_2 \approx \mathbb{Z}^{20} & H_2(L, M) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{280} / \mathbb{Z}^{280} = 0 \\ \ker d_3 \approx \mathbb{Z}^{1545} & \text{im } d_3 \approx \mathbb{Z}^{280} & H_3(L, M) = \ker d_3 / \text{im } d_4 \approx \mathbb{Z}^{1545} / \mathbb{Z}^{1299} \approx \mathbb{Z}^{246} \\ \ker d_4 \approx \mathbb{Z}^{1393} & \text{im } d_4 \approx \mathbb{Z}^{1299} & H_4(L, M) = \ker d_4 / \text{im } d_5 \approx \mathbb{Z}^{1393} / \mathbb{Z}^{1386} \approx \mathbb{Z}^7 \\ \ker d_5 \approx \mathbb{Z}^{924} & \text{im } d_5 \approx \mathbb{Z}^{1386} & H_5(L, M) = \ker d_5 / \text{im } d_6 \approx \mathbb{Z}^{924} / \mathbb{Z}^{924} = 0 \\ \ker d_6 \approx \mathbb{Z}^{396} & \text{im } d_6 \approx \mathbb{Z}^{924} & H_6(L, M) = \ker d_6 / \text{im } d_7 \approx \mathbb{Z}^{396} / \mathbb{Z}^{396} = 0 \\ \ker d_7 \approx \mathbb{Z}^{99} & \text{im } d_7 \approx \mathbb{Z}^{396} & H_7(L, M) = \ker d_7 / \text{im } d_8 \approx \mathbb{Z}^{99} / \mathbb{Z}^{99} = 0 \\ \ker d_8 \approx \mathbb{Z}^{11} & \text{im } d_8 \approx \mathbb{Z}^{99} & H_8(L, M) = \ker d_8 / \text{im } d_9 \approx \mathbb{Z}^{11} / \mathbb{Z}^{11} = 0 \\ \ker d_9 = 0 & \text{im } d_9 \approx \mathbb{Z}^{11} & H_9(L, M) = \ker d_9 / \text{im } d_{10} = 0/0 = 0 \\ \ker d_i = 0 & \text{im } d_i = 0 & H_i(N) = \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 10 \end{array}$$

$N = \mathbf{N}(7, 3, 0) \approx \mathbf{N}(7, 4, 1)$  (which is not 1-connected)

$$\begin{array}{lll} \ker d_0 \approx \mathbb{Z}^{35} & \text{im } d_0 = 0 & H_0(N) = \ker d_0 / \text{im } d_1 \approx \mathbb{Z}^{35} / \mathbb{Z}^{34} \approx \mathbb{Z} \\ \ker d_1 \approx \mathbb{Z}^{176} & \text{im } d_1 \approx \mathbb{Z}^{34} & H_1(N) = \ker d_1 / \text{im } d_2 \approx \mathbb{Z}^{176} / \mathbb{Z}^{105} \approx \mathbb{Z}^{71} \\ \ker d_2 \approx \mathbb{Z}^{35} & \text{im } d_2 \approx \mathbb{Z}^{105} & H_2(N) = \ker d_2 / \text{im } d_3 \approx \mathbb{Z}^{35} / \mathbb{Z}^{35} = 0 \\ \ker d_3 = 0 & \text{im } d_3 \approx \mathbb{Z}^{35} & H_3(N) = \ker d_3 / \text{im } d_4 = 0/0 = 0 \\ \ker d_i = 0 & \text{im } d_i = 0 & H_i(N) = \ker d_i / \text{im } d_{i+1} = 0/0 = 0, \text{ for all } i \geq 4 \end{array}$$

## 5.6 Comparison with Upper Bounds for Chromatic Numbers

With the graphs given in the examples in Section 5.5 as well as many others, it is comparatively easy to find the upper bounds for the chromatic numbers. Without much thought at all, an obvious upper bound for any graph would be the total number of vertices in the graph. Yet, with a touch of thought, we can scale this figure down considerably. The next example illustrates this. It will be seen that in most of the cases in Section 5.5 the upper and lower bounds coincide, thus giving us the exact chromatic number. In this section we will only consider the graphs  $G(n, r, q)$  where  $q > 0$  since  $q = 0$  is dealt with in [7]. Lovász's paper actually shows that when  $q = 0$  the upper and lower bounds agree.

**Example 5.6.1** In Section 5.5,  $N(4, 2, 1)$  is 0-connected, which implies that  $\chi(G(4, 2, 1)) \geq 0 + 3 = 3$ . Here is an explicit 3-colouring for  $G(4, 2, 1)$ .

Blue	Navy	Cyan
{1, 2}	{1, 3}	{1, 4}
{3, 4}	{2, 4}	{2, 3}

Therefore an upper bound for the the chromatic number of  $G(4, 2, 1)$  is 3, which is the same as its calculated lower bound, implying that the chromatic number for  $G(4, 2, 1)$  is precisely 3.

**Example 5.6.2** From the data in Section 5.5 one concludes that  $N(5, 2, 1)$  is 2-connected which implies that  $\chi(G(5, 2, 1)) \geq 2 + 3 = 5$ . Here is an explicit 5-colouring of  $G(5, 2, 1)$ .

Green	Turquoise	Chartreuse	Avocado Green	Forest Green
{1, 2}	{1, 3}	{1, 4}	{1, 5}	{2, 3}
{3, 4}	{2, 5}	{3, 5}	{2, 4}	{4, 5}

Thus in this case both the upper and lower bound are the same, implying that the chromatic number of  $G(5, 2, 1)$  is in fact 5.

**Example 5.6.3** Recall that in Section 5.5 it is shown (also using the Hurewicz Theorem 4.3.4 and Theorem 5.3.7) that  $N(6, 2, 1)$  is 2-connected, which implies by the Lovász Theorem, Theorem 4.3.10, that  $\chi(G(6, 2, 1)) \geq 2 + 3 = 5$ . In fact, here is an explicit 5-colouring for  $G(6, 2, 1)$ .

Maroon	Green	Gold	Silver	Blue
{1, 2}	{1, 3}	{1, 4}	{1, 5}	{1, 6}
{3, 4}	{2, 5}	{2, 6}	{2, 4}	{2, 3}
{5, 6}	{4, 6}	{3, 5}	{3, 6}	{4, 5}

**Example 5.6.4** Recall that from Section 5.5,  $N(7, 2, 1)$  is 2-connected. Therefore,  $\chi(G(7, 2, 1)) \geq 2 + 3 = 5$ . In this particular case, the upper bound that we have found is not 5. Here is an explicit 7-colouring of  $G(7, 2, 1)$ .

Purple	Violet	Grape	Indigo	Plum	Lavender	Mauve
{1, 2}	{1, 3}	{1, 4}	{1, 5}	{1, 6}	{1, 7}	{2, 7}
{3, 4}	{2, 5}	{2, 3}	{2, 6}	{2, 4}	{3, 5}	{3, 6}
{5, 6}	{4, 7}	{6, 7}	{3, 7}	{5, 7}	{4, 6}	{4, 5}

Therefore  $5 \leq \chi(G(7, 2, 1)) \leq 7$ .

**Example 5.6.5** Note that the neighbourhood simplicial complexes for both  $N(6, 3, 1)$  and  $N(6, 3, 2)$  are identical. In fact, the graphs  $G(6, 3, 1)$  and  $G(6, 3, 2)$  are isomorphic. This does not follow from duality or seem to follow from any other general principle, but we will show it by exhibiting a specific isomorphism.

Let  $\phi : V \rightarrow V'$  where  $G(6, 3, 1) = (V, E)$  and  $G(6, 3, 2) = (V', E')$ . We will define  $\phi$  as follows.

$v$	$\phi(v)$	$v$	$\phi(v)$	$v$	$\phi(v)$	$v$	$\phi(v)$
{1, 2, 3}	{1, 2, 3}	{1, 3, 5}	{1, 4, 6}	{2, 3, 4}	{2, 4, 6}	{2, 5, 6}	{1, 2, 6}
{1, 2, 4}	{1, 5, 6}	{1, 3, 6}	{2, 5, 6}	{2, 3, 5}	{3, 5, 6}	{3, 4, 5}	{1, 2, 5}
{1, 2, 5}	{2, 4, 5}	{1, 4, 5}	{2, 3, 6}	{2, 3, 6}	{1, 4, 5}	{3, 4, 6}	{1, 3, 6}
{1, 2, 6}	{3, 4, 6}	{1, 4, 6}	{1, 2, 4}	{2, 4, 5}	{1, 3, 4}	{3, 5, 6}	{2, 3, 4}
{1, 3, 4}	{3, 4, 5}	{1, 5, 6}	{1, 3, 5}	{2, 4, 6}	{2, 3, 5}	{4, 5, 6}	{4, 5, 6}

One can tediously check that  $\phi$  is an isomorphism of graphs. Thus we will deal with  $G(6, 3, 1)$  only and  $G(6, 3, 2)$  will reap the benefits directly from this calculation. From Section 5.5 we have discovered that  $N(6, 3, 1)$  is 1-connected. Therefore,  $\chi(G(6, 3, 1)) \geq 1 + 3 = 4$ . Here is an explicit 6-colouring of the  $G(6, 3, 1)$ .

Red	Pink	Maroon	Ruby	Magenta	Crimson
{1, 2, 3}	{1, 3, 4}	{1, 4, 5}	{2, 3, 4}	{2, 4, 5}	{3, 4, 5}
{1, 2, 4}	{1, 3, 5}	{1, 4, 6}	{2, 3, 5}	{2, 4, 6}	{3, 4, 6}
{1, 2, 5}	{1, 3, 6}	{1, 5, 6}	{2, 3, 6}	{2, 5, 6}	{3, 5, 6}
{1, 2, 6}					{4, 5, 6}

Therefore  $4 \leq \chi(G(6, 3, 1)) \leq 6$ . Similarly for  $G(6, 3, 2)$ .



## 5.7 Checking Procedures

The program was written in Maple to create the differential matrices for a given simplicial complex  $N(n, r, q)$ . The data generated by the homology program was checked by the running the relative homology program, comparing the resulting data, and scrutinizing the output before the relative homology program was created. In order to check the relative homology program a completely different algorithm was used. The programming language C was chosen to generate files and compare them to the ones generated by Maple. That way if there were any problems in the Maple program it would have become clear with this new algorithm. First, the highest nonempty  $n$ -chains generated for both  $K$  and  $M$  from the relative homology program were checked by hand. A procedure was created to make sure that the dimension  $n$  flag matrices did not share hash values, which would mean that they would either have a basis element in common or the hash function was not working properly. After establishing that the bases for  $C_n(K)$  and  $C_n(M)$  were disjoint a hash table file for each  $\binom{n}{r}$  case was created. This hash table did not use the hash function. Each row in a hash table file would consist of the  $r$ -subset of  $\underline{n}$  in a lexicographical order and the last integer in the row would represent the row number. In fact this row number is precisely the hash value for the  $r$ -subset. Using the dimension  $n$  flag matrices the files representing both  $C_n(K)$  and  $C_n(M)$  were generated. Comparing these files with the ones that Maple had created enabled us to see if the hash function and flag matrix function were working properly. The data in each case should have been identical, otherwise the hash function was not working properly. After successfully passing these series of tests, it was obvious that the hash function was working

properly, since we could create  $C_n(K)$  and  $C_n(M)$  from the flag matrices. Thus the differential files were also created using both the *pos.n* and *neg.n* files to make sure that  $\pm 1$  and 0's were placed in the correct positions in the differential files. Some of the smaller differential files were checked by recreating the differential files with the C code and comparing them to the ones generated by the Maple program.

## Appendix A

### Homology Program

The homology program described in this appendix is written in MAPLE, and is specifically geared to calculate the differential matrices of the neighbourhood complex of the generalized Kneser graphs  $G(n, r, q)$ . Each procedure in the program will be described either in this appendix or in the following one.

**change** requires the input  $A$  which must be a matrix and a string of text which is the filename the user wants the information to be stored under in their directory. This procedure first opens up the file (which it is going to write to) and enters one number per line. The first number is the number of rows in the matrix  $A$ . The second number is the number of columns in  $A$ . Then for every line it enters an element of the matrix, and it reads the matrix row by row. For example for the matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

typing `change(A,first-try)`; in MAPLE will produce the file called `first-try` and in that file the information will be displayed as:

```
2
3
1
2
3
4
5
6
```

The reason I have this procedure is to run Arne Storjohann's *Integral Smith* program, which can handle larger matrices and with more efficiency than MAPLE.

```
change := proc(A,filename)
  local i,j,m,n;
  writeto(filename);
  m := rowdim(A);
```

```

n := coldim(A);
  lprint(m);
  lprint(n);

  for i to m do
    for j to n do lprint(A[i,j]); od;
  od;
  writeto(terminal);
end;

```

*See Appendix B for a description.*

```

binsearch := proc(value,A,top,bottom)
  local compare,leftover,mid,i;
  compare := bottom + top;
  leftover := modp(compare,2);
  mid := (compare + leftover)/2;
  compare := A[mid,1];

  if (A[1,1] = value) then
    RETURN(eval(A[1,2]));
  fi;

  if compare < value then i := binsearch(value,A,mid,bottom); fi;

  if compare > value then i := binsearch(value,A,top,mid); fi;

  if (compare = value) then RETURN(eval(A[mid,2])); fi;

  RETURN(eval(i));
end;

```

*See Appendix B for a description.*

```

differential := proc(rows,cols,iplace)
  local i,j,p,n,nump,numn,ncou,pcou,Pos,Neg;
  interface(quiet = true);

```

```

print('in differential, iplace is: reading pos and neg',iplace);

nump := readdata(pos.iplace,integer,2);
numn := readdata(neg.iplace,integer,2);

Pos := convert(nump,matrix);
Neg := convert(numn,matrix);

writeto(differential.iplace);
nump := rowdim(Pos);
numn := rowdim(Neg);
ncou := 1;
pcou := 1;
p := Pos[1,1];
n := Neg[1,1];

lprint(rows);
lprint(cols);

for i from 1 to rows do
  for j from 1 to cols do
    if (((i - 1) * cols + j) = p) then
      lprint(1,((i - 1) * cols) + j,pcou);
      pcou := pcou + 1;
      if pcou <= nump then p := Pos[pcou,1]; fi;
    elif (((i - 1) * cols + j) = n) and (ncou <= numn) then
      lprint(-1,((i - 1) * cols) + j,ncou);
      ncou := ncou + 1;
      if ncou <= numn then n := Neg[ncou,1]; fi;
    else lprint(0);
      fi;
    od;
  od;
end;

```

*See Appendix B for a description for both partition and qsort.*

```

partition := proc(A,x,y,i)
  local m,k,left,right, dummy,pivot;

```

```

m := y + x;
if modp(m,2) = 0 then k := m/2
else k := (m + 1)/2;
fi;

pivot := A[k,i];
A :=swaprow(A,x,k);

left := x;
right := y + 1;
dummy := true;

while dummy do
  while ( l = 1 ) do
    right := right - 1;
    if A[right,i] <= pivot then break fi;
    if right = x then break fi;
  od;

  while ( l = 1 ) do
    left := left + 1;
    if A[left,i] >= pivot then break fi;
    if left = y then break fi;
  od;

  if left < right then A := swaprow(A,left,right);
  else dummy := false;
    A := swaprow(A,x,right); fi;
od;

if left >= right then RETURN(eval(right)); fi;
end;

```

```

qsort := proc(A,x,y,i)
  local q,temp,m;
  m := rowdim(A);
  q := partition(A,x,y,i);
  if q > x + 1 then qsort(A,x,q,i); fi;

```

```

    if q < y - 1 then qsort(A,q + 1,y,i); fi;
end;

```

*See Appendix B for a description.*

```

repetition := proc(A)
    local B,C;
    B := convert(A,listlist);
    C := convert(B,set);
    B := convert(C,list);
    C := convert(B,listlist);
    B := convert(C,matrix);
    RETURN(eval(B));
end;

```

*See Appendix B for a description.*

```

hash := proc(row1,length,numVert)
    local amount,sum,i,first,second,chosen,j;
    amount := vectdim(row1);
    sum := 1;

    if row1[1] > 1 then
        for j to (row1[1] - 1) do
            chosen := numbcmb((numVert - j),(length - 1));
            sum := sum + chosen;
        od;
    fi;

    for i from 2 to amount do
        first := row1[i - 1] + 1;
        second := row1[i] - 1;

        if second - first > -1 then
            for j from first to second do
                chosen := numbcmb((numVert - j),(length - i));
                sum := sum + chosen;
            od;
        fi;
    od;
end;

```

```

        od;
    fi;
od;

RETURN(eval(sum));
end;

```

*See Appendix B for a description.*

```

createflag := proc(rowA,colA,iplace,verticies,n,r,q)
    local i,j,A,first,place,size,flag,leftover,counter,mult;
    print('Verticies',verticies);
    size := numbcomb(verticies,colA);
    flag := array(1..rowA,1..2);
    counter := 0;

    leftover := modp(rowA,20);

    if rowA >= 20 then mult := (rowA - leftover)/20
    else mult := 0;
    fi;

    for j from 1 to mult do
        A := readdata(Ci.n.r.q.iplace.j,integer,colA);

        for i to 20 do
            first := A[i];
            size := convert(first,list);
            first := convert(size,vector);
            place := hash(first,colA,verticies);
            flag[counter+i,1] := place;
            flag[counter+i,2] := counter + i;
        od;

        counter := counter + 20;
    od;

    if 0 < leftover then
        mult := mult + 1;
    fi;
end;

```



```

A := readdata(Ci.n.r.q.iplace.mult,integer,colA);

for i from (counter + 1) to rowA do
  first := A[i - counter];
  j := convert(first,list);
  first := convert(j,vector);
  place := hash(first,colA,verticies);
  flag[i,1] := place;
  flag[i,2] := i;
od;
fi;

if rowdim(flag) < 50 then
  print('the flag for C_',iplace,'is',flag);
fi;

qsort(flag,1,rowA,1);
writedata(flag.iplace,flag,integer);
flag := 'flag';
A := 'A';
end;

```

*See Appendix B for a description.*

```

pn := proc(numEdges,numVerticies,edgeLength,ione,n,r,q,m)
  local leftover,i,j,multiples,A,counter,k,B,edge,Edge,u,newedge,place,
    spot,found,numNegrow,numPosrow,pos,neg,pcou,ncou,flag,
    VertexLength,Pos,Neg,c;
  interface(quiet = true);
  leftover := modp(numEdges,20);
  k := ione - 1;
  VertexLength := edgeLength - 1;
  pcou := readdata(flag.k,integer,2);
  flag := convert(pcou,matrix);
  pcou := 1;
  ncou := 1;
  k := modp(edgeLength,2);
  numNegrow := ((edgeLength - k)/2) * numEdges;
  numPosrow := ((edgeLength + k)/2) * numEdges;

```

```

Pos := matrix(numPosrow,1);
Neg := matrix(numNegrow,1);

if numEdges >= 20 then multiples := (numEdges-leftover)/20;
else multiples := 0;
fi;

counter := 0;

for j from 1 to multiples do
  A := readdata(Ci.n.r.q.ione.j,integer,edgeLength);

  for k to 20 do
    c := convert(A[k],list);
    B := convert(c,vector);
    edge := convert(B,matrix);

    for u to edgeLength do
      newedge := delrows(edge,u..u);
      Edge := convert(newedge,vector);
      place := hash(Edge,VertexLength,m);
      spot := binsearch(place,flag,1,numVertices);

      if spot = 0 then
        print('Hash is not working properly');
        found := false;
      else
        if modp(u,2)=1 then
          Pos[pcou,1]
            := (((counter + k - 1) * (numVertices)) + spot);
          pcou := pcou + 1;
        else
          Neg[ncou,1]
            := (((counter + k - 1) * (numVertices)) + spot);
          ncou := ncou + 1;
        fi;
      fi;
    od;
  od;
od;

```

```

    counter := counter + 20;
od;

if 0 < leftover then
    multiples := multiples + 1;
    print('multiples = ',multiples);
    print('ci.n.r.q.ione',n,r,q,ione,multiples);
    A := readdata(Ci.n.r.q.ione.multiples,integer,edgeLength);

    for k from (counter + 1) to numEdges do
        c := convert(A[k - counter],list);
        B := convert(c,vector);
        edge := convert(B,matrix);

        for u to edgeLength do
            newedge := delrows(edge,u..u);
            Edge := convert(newedge,vector);
            place := hash(Edge,VertexLength,m);
            spot := binsearch(place,flag,1,numVerticies);
            print('row,place ,spot',Edge,place,spot);

            if spot = 0 then
                appendto(die.ione);
                print('Hash is not working properly');
                found := false;
                writeto(terminal);
            else
                if modp(u,2) = 1 then
                    Pos[pcou,1] := (((k - 1) * numVerticies) + spot);
                    pcou := pcou + 1;
                else Neg[ncou,1] := (((k - 1) * numVerticies) + spot);
                    ncou := ncou + 1;
                fi;
            fi;
        od;
    od;
od;

fi;
qsort(Pos,1,numPosrow,1);
qsort(Neg,1,numNegrow,1);
writedata(pos.ione,Pos,integer);

```

```
writedata(neg.ione,Neg,integer);
print('Dimensions of d',ione, 'are',numEdges,'by',numVertices);
end;
```

*See Appendix B for a description.*

```
knCI := proc(maxsimps,i,n,r,q)
  local amount, j, templ, isubset,B,A,columns;
  amount := vectdim(maxsimps);
  for j to amount do
    templ := maxsimps[j];
    isubset := choose(templ, i + 1);
    templ := convert(isubset,listlist);
    isubset := convert(templ,matrix);
    templ := 0;

    if j = 1 then A := copy(isubset);
    else B := stack(A,isubset);
      A := copy(B);
      B := 0;
    fi;

    isubset := 0;
    templ := 0;
  od;

  B := repetition(A);
  writedata(allci.n.r.q.i,B,integer);
  templ := rowdim(B);
  columns := coldim(B); isubset := modp(templ,20);

  if templ >= 20 then amount := (templ - isubset)/20;
  else amount := 0;
  fi;

  for j from 1 to amount do
    A := submatrix(B,(((j - 1) * 20) + 1)..(j * 20),1..columns);
    writedata(Ci.n.r.q.i.j,A,integer);
  od;
```

```

if isubset > 0 then
  A := submatrix(B,(temp1 - isubset + 1)..temp1,1..columns);
  amount := amount + 1;
  writedata(Ci.n.r.q.i.amount,A,integer);
fi;

print('There are ', temp1 , 'elements in C_',i);
A := [temp1,columns];
RETURN(eval(A));
end;

```

*See Appendix B for a description.*

```

knhomology := proc(Simplexes,total,r,inter)
  local noMaxSimplexes, LargestSimplex, a,A,size,maxsimp,b,B,c,C,t,
        Ci1,q,D,i,j,Ci,flag,mi,counter,m,v,w,n,rowCi,colCi,numEdges,
        lengthEdges,verticies;
  verticies := numcomb(total,r);
  noMaxSimplexes := vectdim(Simplexes);
  LargestSimplex := 1;

  for a to noMaxSimplexes do
    A := convert(Simplexes[a],list);
    size := vectdim(A);
    if LargestSimplex < size then LargestSimplex := size fi;
  od;

  for b from LargestSimplex by -1 to 2 do
    appendto(kn.total.r.inter);
    if b = LargestSimplex then
      A := knCI(Simplexes,b-1,total,r,inter);
      numEdges := A[1];
      lengthEdges := A[2];
    fi;

    A := knCI(Simplexes,b - 2,total,r,inter);
    rowCi := A[1];
    colCi := A[2];
  end;

```

```

    flag := [ ];
    createflag(rowCi,colCi,b - 2,vertices,total,r,inter);
    pn(numEdges,rowCi,lengthEdges,b - 1,total,r,inter,vertices);
    differential(numEdges,rowCi,b - 1);
    if b > 2 then
        numEdges := rowCi;
        lengthEdges := colCi
    fi;

    flag := [ ];
    q := [ ];
    A := [ ];
    gc();
od;
end;

```

*See Appendix B for a description.*

```

knngbdsimplex := proc(MaximalSimplexes,n,r,q)
    local amount, counter, t,i,j,u,v,vertices,temp,size,big,l,A,B,C,k,D;
    amount := vectdim(MaximalSimplexes);
    big := 0;

    for i to amount do
        u := MaximalSimplexes[i];
        v := convert(u,list);
        size := vectdim(v);
        big := big + size;
    od;

    temp := matrix(big,1,0);
    counter := 0;
    for i to amount do
        u := MaximalSimplexes[i];
        v := convert(u,list);
        size := vectdim(v);

        for j to size do
            temp[counter + j,1] := v[j];

```

```

    od;

    counter := counter + size;
od;

vertices := repetition(temp);
counter := rowdim(vertices);
temp := array(1..counter);
for i to counter do
    t := 0;
    A := array(1..big);

    for j to amount do
        u := MaximalSimplexes[j];
        v := convert(u,vector);
        u := convert(v,matrix);
        size := rowdim(u);

        for k to size do
            if vertices[i,1] = v[k] then
                C := delrows(u,k..k);

                for l to (size - 1) do A[l+t] := C[l,1]; od;

                t := t + size - 1;
                break;
            fi;
        od;
    od;
    B := matrix(t,1);

    for l to t do B[l,1] := A[l] od;

    A := repetition(B);
    B := transpose(A);
    C := convert(B,set);
    D := convert(C,list);
    temp[i] := D;
od;
appendto(kn.n.r.q);

```

```

    print('Neighbourhood Maximal Simplexes',temp);
    RETURN(eval(temp));
end;

```

*See Appendix B for a description.*

```

knngbdcomplex := proc(MaximalSimplexes,n,r,q)
    local A,B,C;
    A := copy(MaximalSimplexes);
    B := knngbdsimplex(A,n,r,q);
    A := [ ];
    knhomology(B,n,r,q);
end;

```

*See Appendix B for a description.*

```

kneser := proc(n,r,q)
    local Points, temp, numberOfPoints,i,j,t,Edges,ipoint,jpoint,
        common,size,flag,A,B,C;
    Points := choose(n,r);
    numberOfPoints := nops(Points);
    B := matrix(1,2,0);
    Edges := matrix(1,2,0);
    flag := 0;
    writeto(vert.n.r.q);
    print('knland7.ms: The verticies are: ',Points);
    writeto(terminal);
    for i to (numberOfPoints - 1) do
        A := matrix(numberOfPoints - i, 2, 0);
        t := 1;
        ipoint := convert(Points[i],set);

        for j from i + 1 to numberOfPoints do
            jpoint := convert(Points[j],set);
            common := ipoint intersect jpoint;
            size := nops(common);

            if size = q then

```



```

        A[t,1] := i;
        A[t,2] := j;
        t := t + 1;
        flag := flag + 1;
    fi;
od;

if (1 < t) and (t < numberOfPoints - i + 1) then
    C := delrows(A,t..numberOfPoints - i);
    Edges := stack(C,B);
else
    if flag > 0 and t > 1 then
        Edges := stack(A,B);
    else Edges := B;
    fi;

fi;

if t = 1 and flag = 0 then
    print('There are no edges from the vertex', ipoint,
        'to the other vertices in the graph');
fi;

temp := rowdim(Edges);
if (i = 1) and (t <> 1) then
    A := delrows(Edges,temp..temp);
    Edges := copy(A);
fi;

B := copy(Edges);
od;

temp := convert(Edges,listlist);
Edges := convert(temp,list);
RETURN(eval(Edges));
end;
```

*See Appendix B for a description.*

```

NOEDGES := proc(n,r,q)
  local Points, temp, numberOfPoints,i,j,t,ipoint,jpoint,common,size,Edges;
  Points := choose(n,r);
  numberOfPoints := nops(Points);
  t := 1;

  for i to (numberOfPoints - 1) do
    ipoint := convert(Points[i],set);

    for j from i + 1 to numberOfPoints do
      jpoint := convert(Points[j],set);
      common := ipoint intersect jpoint;
      size := nops(common);

      if size = q then
        t := 2;
        break;
      fi;
    od;
  od;

  RETURN(evalb(t = 1));
end;

```

*See Appendix B for a description.*

```

knNgbdcomplex := proc(n,r,q)
  local kn,p;

  if NOEDGES(n,r,q) then
    writeto(kn.n.r.q);
    kn := kneser(n,r,q);
    writeto(terminal);
  else kn := kneser(n,r,q);
    writeto(kn.n.r.q);
    print('The edges in the kneser graph G',n,r,q,'are',kn);
    writeto(terminal);
  end;

```

```
        kngbdcomplex(kn,n,r,q);  
    fi;  
    writeto('terminal');  
end;
```

## Appendix B

### Maple Program of the Relative Homology

This program is different from the previous program (Appendix A) due to the fact that it calculates a relative homology of the neighbourhood simplicial complex of the Kneser graph,  $G(n, r, q)$ . First it will calculate the star of a vertex, call this star  $M$ , and then determine  $C_*(L, M)$  where  $L$  is the neighbourhood complex. It is also more dependent on the procedure `pn` to help create the differential matrices. The reason for the change in the program, is that with the calculation of the relative homology, the size of the differential matrices is smaller or the same size as the differential matrices in the calculation of the neighbourhood complex itself. This allows for quicker results.

Before we start the documentation of the relative homology program, we will need a useful definition first.

**Definition B.0.1** *Let  $K = (V, \Sigma)$  be a simplicial complex. The star of a vertex  $v \in V$  is the subcomplex of all simplexes  $\sigma \subseteq \tau$ , for some  $\tau \in \Sigma$  such that  $v \in \tau$ , and is denoted  $star(v)$  (also see Definition 5.4.2 on page 90).*

Also notice that  $star(v)$  is contractible for any  $v \in V$ , cf. 5.4.5 on page 91.

**Notation B.0.2**  $L'$  represents either the simplicial complex  $K$  or  $M$ . The comments for each procedure are in italics. The variables for that particular procedure are in bold face as well as the names of all the procedures which appear in the comments.

The function *binsearch* is a search engine used to find entries in the flag matrices. *binsearch* is a binary recursive search function. It assumes that the first column of matrix **A** is in increasing order, that is the  $A[i,1] \leq A[i+1,1]$  for all  $1 \leq i \leq m-1$  where  $m$  is the number of rows in **A**. If *value* is found then *binsearch* returns the value in the second column of the corresponding row. That is, if  $A[j,1] = \text{value}$ , then *binsearch* will return  $A[j,2]$ , for some  $1 \leq j \leq m$ . Otherwise *binsearch* will return  $-1$ .

**A** is an array that consists of at least two columns such that the first column is in increasing order.

*value* is a number to be found in the first column of **A**.

*top* is an integer, which represents the row to start the binary search in matrix **A**. Usually when calling the function *top* is assigned the value 1.

*bottom* is an integer which represents the row to end the binary search in matrix **A**. Usually when calling the function *bottom* is assigned the number of rows in **A**.

*oldmid* is an integer, which helps determine if *value* has been found. When calling the function *oldmid* is assigned a non-positive value.

```

binsearch := proc(value,A,top,bottom,oldmid)
  local compare,leftover,mid,i;
  compare := bottom + top;
  leftover := modp(compare,2);
  mid := (compare + leftover)/2;
  compare := A[mid,1];
  if mid = oldmid then RETURN(-1) fi;
  if bottom - top <= 0 then RETURN(-1) fi;
  if (A[1,1] = value) then RETURN(eval(A[1,2])); fi;
  if compare < value then i := binsearch(value,A,mid,bottom,mid); fi;
  if compare > value then i := binsearch(value,A,top,mid,mid); fi;
  if (compare = value) then RETURN(eval(A[mid,2])); fi;
  RETURN(eval(i));
end;

```

The function *repetition* will remove any repeated rows in matrix **A**. That is if  $A[i,k] = A[j,k]$  for some  $1 \leq i \neq j \leq m$  and for all  $1 \leq k \leq n$  where  $m$  is

the number of rows of  $A$  and  $n$  is the number of columns, either the  $i$ th or  $j$ th row will be deleted. The purpose of **repetition** is to remove redundant elements in the calculations of  $C_i(K)$  and  $C_i(M)$  for some nonnegative integer  $i$ .

$A$  a  $m \times n$  matrix.

```
repetition := proc(A)
  local B,C;
  B := convert(A,listlist);
  C := convert(B,set);
  B := convert(C,list);
  C := convert(B,listlist);
  B := convert(C,matrix);
  RETURN(eval(B));
end;
```

The procedure **qsort** takes a matrix  $A$  and arranges the rows in increasing order with respect to the elements in the  $i$ th column. The purpose of **qsort** is to organize lists of data. The procedure **qsort** is the standard recursive quick sort taught in most introductory level computer programming courses. The function **partition** is the main engine driving **qsort** to work. When calling these two procedures, one just calls **qsort**.

$A$  is an  $m \times n$  matrix where  $m > 1$  and  $A[j, k] \in \mathbb{R}$  for all  $1 \leq j \leq m$  and for all  $1 \leq k \leq n$ .

$x$  is a natural number indicating the first row to start sorting.

$y$  is a natural number indicating the last row to end sorting. and  $x \leq y$ .

$i$  is a natural number indicating the column of the matrix to sort with respect to.

```
partition := proc(A,x,y,i)
  local m,k,left,right, dummy,pivot;
  m := y + x;
  if modp(m,2) = 0 then k := m/2
  else k := (m + 1)/2;
  fi;
  pivot := A[k,i];
  A := swaprow(A,x,k);
```

```

left := x;
right := y + 1;
dummy := true;
while dummy do
  while (l=1) do
    right := right - 1;
    if A[right,i] <= pivot then break fi;
    if right = x then break fi;
  od;
  while (l=1) do
    left := left + 1;
    if A[left,i] >= pivot then break fi;
    if left = y then break fi;
  od;
  if left < right then A := swaprow(A,left,right);
  else dummy := false;
    A := swaprow(A,x,right);
  fi;
od;
if left >= right then RETURN(eval(right)); fi;
end;

qsort := proc(A,x,y,i)
  local q,temp,m;
  m := rowdim(A);
  q := partition(A,x,y,i);
  if q > x + 1 then qsort(A,x,q,i); fi;
  if q < y - 1 then qsort(A,q+1,y,i); fi;
end;

```

*The function **star** returns a list of all maximal simplexes in the star of a given vertex. The type returned is a list of lists. The purpose of **star** is to calculate the star of a vertex. This will help reduce the time required to compute the Integral Smith Normal Form of the differential matrices.*

***L** is a list of lists representing all the maximal simplexes in the complex.*

*vertex is an element of the elements of **L**, i.e. if  $l \in L$  then vertex may be an element in  $l$ .*

**no\_maxsimps** represents the number of maximal simplexes **L**, that is the cardinality of **L**.

```

star := proc(vertex,L,no_maxsimps)
  local i,M,counter,maximalsimplex,temp;
  counter := 1;
  M := array(1..no_maxsimps);
  for i to no_maxsimps do
    temp := L[i];
    maximalsimplex := convert(temp,list);
    if (member(vertex,maximalsimplex)) then
      M[counter] := maximalsimplex;
      counter := counter + 1;
    fi;
  od;
  if (counter < no_maxsimps) then
    temp := convert(M,set);
    M := convert(temp,list);
  fi;
  print('in star and this is what the star of',vertex,':',M);
  RETURN(eval(M));
end;

```

The hash function is a one-to-one map which takes **row1** and determines its position in a complete lexicographically ordered set containing sets with the same cardinality, varying entries and natural ordering of the entries. The purpose of **hash** is to improve the efficiency of the program with respect to comparing two vectors/lists or searching for one.

**row1** is a vector or a list of natural numbers.

**length** is the number of entries in **row1**.

**numVert** is a natural number representing the total number of vertices in the graph or neighbourhood complex.

```

hash := proc(row1,length,numVert)
  local amount,sum,i,first,second,chosen,j;
  amount := vectdim(row1);
  sum := 1;

```



```

if row1[1] > 1 then
  for j to (row1[1] - 1) do
    chosen := numcomb((numVert - j),(length - 1));
    sum := sum + chosen;
  od;
fi;
for i from 2 to amount do
  first := row1[i - 1] + 1;
  second := row1[i] - 1;
  if second - first > -1 then
    for j from first to second do
      chosen := numcomb((numVert - j),(length - i));
      sum := sum + chosen;
    od;
  fi;
od;
RETURN(eval(sum));
end;

```

The function **createflag** assumes there are certain files in the current directory which start with **name**. These files can be created using **knCi** and **write20** and contain fragments of matrix *A*. The function creates a flag matrix for *A*. That is, it will create a flag matrix for the elements of  $C_{iplace}(L)$ . The first column of the matrix represents the hash values for each row in *A* or for each element of  $C_{iplace}$ . The second column represents the row they appear in *A*. The purpose of **hash** is to increase the speed of both comparing and searching for elements of  $C_{iplace}$ .

**rowA** is a natural number which indicates the number of rows in matrix *A*.

**iplace** is a non-negative integer indicating the *i*-cell or  $C_{iplace}$ .

**name** is a string of characters indicating the partial name of the files to use in the calculation of the flag matrix for *A*.

**vertices** is a natural number representing the number of vertices in the graph or neighbourhood complex.

**n**, **r** are natural numbers which are represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

**q** is a nonnegative integer which is represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

```

createflag := proc(rowA,iplace,name,vertices,n,r,q)
  local i,j,A,first,place,size,flag,flagsize,leftover,counter, mult,hashvalue,colA,
    extra,top,bottom;
  print('Vertices', vertices);
  print('Name is', name);
  appendto(kn.n.r.q);
  colA := iplace + 1;
  size := rowA;
  flag := matrix(size,2,0);
  leftover := modp(rowA,20);
  top := 1;
  if rowA >= 20 then
    mult := ((rowA - leftover)/20);
    bottom := 20;
  else mult := 0;
    bottom := rowA;
  fi;
  if leftover > 0 then mult := mult + 1; fi;
  for j from 1 to mult do
    A := readdata(name.n.r.q.iplace.j,integer,colA);
    extra := (j - 1) * 20;
    for i from top to bottom do
      first := A[i - extra];
      size := convert(first,list);
      first := convert(size,vector);
      place := hash(first,colA,vertices);
      if flag[i,1] = 0 then
        flag[i,1] := place;
        flag[i,2] := i;
      else appendto(problem);
        print('CRASH, ERROR IN createflag');
        print('Variable name is: ',name);
        print('Variable iplace is: ',iplace);
        print('counter and i are: ',counter,i);
        print('n,r,q are: ',n,r,q);
        writeto(terminal);
      fi;
    od;
    top := top + 20;
    if j = (mult - 1) then bottom := rowA

```

```

        else bottom := bottom + 20;
        fi;
    od;
    if rowdim(flag) > 1 then
        if rowdim(flag) < 50 then print('the flag for ', name, '- ', iplace, flag); fi;
        qsort(flag,1,rowA,1);
    fi;
    RETURN(eval(flag));
end;
```

The function **matrixchoose** returns a matrix with all the **choosesize** distinct subsets of **alist[elementnum]**.

**alist** is a list of lists/vectors of natural numbers.

**elementnum** is a natural number which represents an entry position in **alist**.

**choosesize** is a natural number which represents the size of subsets of **alist[elementnum]** to calculate.

```

matrixchoose := proc(alist,elementnum,choosesize)
    local temp,A;
    temp := alist[elementnum];
    A := choose(temp, choosesize);
    temp := convert(A,listlist);
    A := convert(temp,matrix);
    RETURN(eval(A));
end;
```

The procedure **write20** takes the matrix **A** and creates a series of files with the filename: **name.n.r.q.iplace.j**. Let **rows** be the number of rows in **A** and  $\text{row20} = \text{rows} \bmod 20$ . Then  $1 \leq j \leq (\text{rows} - \text{row20})/20 + 1$ . Please note that the '.' does not appear in the filename, it is used to separate the variables and make it easier to read. Also note that the values of the variables appear in the filename. For example, let **name** = 'Mi', **n** = 5, **r** = 2, **q** = 0, **iplace** = 1 and **j** = 1. Then the filename create would be **Mi52011**. The purpose of **write20** is to speed things up by using more storage memory and less "active" memory when calculating the flag matrices.

**A** is a matrix.

**name** is a string of characters representing the initial part of the name of the files to be generated.

**iplace** is a nonnegative integer which represents the **iplace** cell, which **A** represents, i.e.  $C_{\text{iplace}}(L') = \mathbf{A}$ .

**n**, **r** are natural numbers which are represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

**q** is a nonnegative number which is represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

```

write20 := proc(A,name,iplace,n,r,q)
  local rows,cols,leftover,amount,B,j;
  rows := rowdim(A);
  cols := coldim(A);
  leftover:= modp(rows,20);
  if rows >= 20 then amount := (rows - leftover)/20;
  else amount := 0;
  fi;
  for j from 1 to amount do
    B := submatrix(A,(((j - 1) * 20) + 1)..(j * 20),1..cols);
    writedata(name.n.r.q.iplace.j,B,integer);
  od;
  if leftover > 0 then
    amount := amount + 1;
    B := submatrix(A,(rows - leftover + 1)..rows,1..cols);
    writedata(name.n.r.q.iplace.amount,B,integer);
  fi;
end;

```

The function **knCI** creates files *all.Ki.n.r.q.i*, *Ki.n.r.q.i.j* or *all.Mi.n.r.q.i*, *Mi.n.r.q.i.j* depending on whether we are determining  $C_i(K)$  or  $C_i(M)$ . Note that *j* depends on the procedure **write20**. The function also returns a two element list,  $[m, n]$  where *m* is the number of rows in  $C_i(K)$  or  $C_i(M)$  and *n* is the number of columns. The purpose of **knCI** is to create  $C_i(K)$  and  $C_i(M)$  and their files.

**maxsimps** is a list of (numerical) lists/vectors which represents the maximal simplices of a complex.

**i** is a non negative integer which represents the *i* value of  $C_i(K)$  or  $C_i(M)$  to be calculated.

**MiB** is a boolean value; *TRUE* if we are to calculate  $C_i(M)$ , and *FALSE* if we are to calculate  $C_i(K)$ .

**flagLength** is a nonnegative integer value which is 0 if we are calculating  $C_i(M)$ .  
Otherwise **flagLength** is the number of rows in the  $C_i(M)$  flag matrix.

**vertices** is a natural number which represents the number of vertices in the graph or neighbourhood complex.

**n**, **r** are natural numbers which are represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

**q** is a nonnegative number which is represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

```

knCI := proc(maxsimps,i,MiB,Miflag,flagLength,vertices,n,r,q)
  local amount,j,temp,ifaces,t,B,A,columns,emptyA, numfaces,numrows,Row,
    hashvalue,face,found,name,leftover;
  amount := vectdim(maxsimps);
  ifaces := matrixchoose(maxsimps,1,i + 1);
  emptyA := 1;
  if MiB then                                MAIN if STATEMENT
    A := copy(ifaces);
    name := 'Mi';
    emptyA := 0;
    for j from 2 to amount do
      ifaces := matrixchoose(maxsimps,j,i + 1);
      temp := stack(A,ifaces);
      A := copy(temp);
    od;
  else
    name := 'Ki';
    numfaces := rowdim(ifaces);
    for j from 2 to amount do
      for t to numfaces do
        face := row(ifaces,t);
        hashvalue := hash(face,i + 1,vertices);
        found := binsearch(hashvalue,Miflag,1,flagLength,0);
        if found = -1 then    not in Mi
          if emptyA = 1 then
            temp := convert(face,listlist);
            A := convert(temp,array);
            temp := convert(A,matrix);
            A := transpose(temp);
          end if;
        end if;
      end for;
    end for;
  end if;
end proc;

```

```

        emptyA := 0;
      else
        temp := stack(A,face);
        A := copy(temp);
      fi;
    fi;
  od;
  ifaces := matrixchoose(maxsimps,j,i + 1);
od;
for t to numfaces do
  face := row(ifaces,t);
  hashvalue := hash(face,i + 1,vertices);
  found := binsearch(hashvalue,Miflag,1,flagLength,0);
  if found = -1 then      not in Mi
    if emptyA = 1 then
      A := convert(face,listlist);
      temp := convert(A,array);
      A := convert(temp,matrix);
    else
      temp := stack(A,face);
      A := copy(temp);
    fi;
  fi;
od;
fi;      MAIN if STATEMENT
if emptyA = 0 then
  B := repetition(A);
  print('The vars of n,r,q,i are:',n,r,q,i);
  writedata(all.name.n.r.q.i,B,integer);
  t := rowdim(B);
  columns := coldim(B);
  write20(B,name,i,n,r,q);
else
  writeto(all.name.n.r.q.i);
  print('empty');
  writeto(terminal);
  appendto(kn.n.r.q);
  t := 0;
  columns := 0;
fi;

```

```

    print('There are ',t,' elements in ',name,i);
    A := [t,columns];
    RETURN(eval(A));
end;

```

*The purpose of the **differential** procedure is to create a file in the proper format compatible with Arne Storjohann's program. This will speed up the calculation of the Integral Smith normal form of the differential matrices. The procedure **differential** assumes that the files **pos.iplace** and **neg.iplace** are in the current running directory. The procedure will create a file which represents the differential matrix in the following format (which is compatible with Arne Storjohann's program).*

```

    rows
    cols
    d1,1
    d1,2
    ⋮
    drows,cols

```

*Please note that if there are other **pos.i** or **neg.i** files in the current directory, for some *i*, this will affect the output of the differential matrix greatly. Since the differential matrix does not require the non-negative integers *n, r, q* of the Kneser graph  $G(n, r, q)$  as input variables to the procedure, it does not have that in the name of the **pos.i** and **neg.i** filenames, which could lead to confusion.*

***rows** is a natural number which represents the number of rows the differential matrix is to have.*

***cols** is a natural number which represents the number of columns the differential matrix is to have.*

***iplace** is a non negative integer which represented in **d<sub>iplace</sub>***

```

differential := proc(rows,cols,iplace)
    local i,j,p,n,nump,numn,ncou,pcou,Pos,Neg,extra;
    interface(quiet=true);
    print('in differential, iplace is: reading pos and neg',iplace);
    nump := readdata(pos.iplace,integer,2);
    numn := readdata(neg.iplace,integer,2);
    Pos := convert(nump,matrix);
    Neg := convert(numn,matrix);

```

```

writeto(differential.iplace);
nump := rowdim(Pos);
numn := rowdim(Neg);
ncou := 1;
pcou := 1;
p := Pos[1,1];
n := Neg[1,1];
lprint(rows);
lprint(cols);
for i from 1 to rows do
  extra := ((i - 1)*cols);
  for j from 1 to cols do
    if ((extra + j) = p) then
      lprint(1);
      pcou := pcou + 1;
      if pcou <= nump then p := Pos[pcou,1]; fi;
    elif ((extra + j) = n) and (ncou <= numn) then
      lprint(-1);
      ncou := ncou + 1;
      if ncou <= numn then n := Neg[ncou,1]; fi;
    else lprint(0);
      fi;
    od;
  od;
writeto(terminal);
end;

```

The procedure `pn` creates two files (1) `pos.ione` and (2) `neg.ione`. The differential matrix  $d_i$  assumes that the rows appear in the same order as the elements of  $C_i(K)$  and its columns are in the same order as the elements of  $C_{i-1}(K)$  in their respective representative matrices. Assuming that the differential matrix is arranged in the similar format as described in the comments of the differential procedure, on page 135 excluding the first two entries (rows and cols), then `pos.ione` contains all the positions of 1 in the differential matrix and `neg.ione` contains all the positions of -1 in the differential matrix. The procedure also requires that `Ki.n.r.q.ione.j` are in the current directory where  $j$  is dependent on the files generated by `write20`. The purpose of `pn` is to save computation time and memory space for the computer. Instead of creating a large zero matrix and then entering  $\pm 1$  in the appropriate entry, which would use a lot of memory space, it just keeps track of all the positions and



saves them into a file which is later accessed by the **differential** procedure and used.

**ione** is a non-negative integer.

**numEdges** is a natural number representing the number of rows in the differential matrix  $D_{\text{ione}}$ .

**numVertices** is a natural number representing the number of columns in the differential matrix  $D_{\text{ione}}$ .

**flag** is a matrix with at least two columns which represents the flag matrix of  $K_{\text{ione}-1}$ .

**n**, **r** are natural numbers which are represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

**q** is a nonnegative number which is represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .

**m** is a natural number which represents the number of vertices in the graph or neighbourhood complex.

```
pn := proc(numEdges,numVertices,flag,ione,n,r,q,m)
  local leftover,i,j,multiples,A,k,B,edge,Edge,u,newedge,place,
  spot,found,numNegrow,numPosrow,pos,neg,pcou,ncou,VertexLength,
  Pos,Neg,c,edgeLength,extra,top,bottom;
  interface(quiet=true);
  leftover := modp(numEdges,20);
  k := ione - 1;
  edgeLength := ione + 1;
  VertexLength := edgeLength - 1;
  if numEdges > 20 then
    multiples := ((numEdges - leftover)/20) + 1;
    bottom := 20;
  else multiples := 1;
    bottom := numEdges;
  fi;
  top := 1;
  pcou := 1;
  ncou := 1;
  k := modp(edgeLength,2);
  numNegrow := ((edgeLength - k)/2) * numEdges;
  numPosrow := ((edgeLength + k)/2) * numEdges;
  print('numNegrow,numPosrow',numNegrow,numPosrow);
  Pos := matrix(numPosrow,1);
  Neg := matrix(numNegrow,1);
```

```

for j from 1 to multiples do
  A := readdata(Ki.n.r.q.ione.j,integer,edgeLength);
  extra := (j - 1) * 20;
  for k from top to bottom do
    c := convert(A[k - extra],list);
    B := convert(c,vector);
    edge := convert(B,matrix);
    for u to edgeLength do
      newedge := delrows(edge,u..u);
      Edge := convert(newedge,vector);
      place := hash(Edge,VertexLength,m);
      spot := binsearch(place,flag,1,numVertices,0);
      if spot = 0 then print('Hash is not working properly');
      elif spot > 0 then
        if modp(u,2) = 1 then
          Pos[pcou,1] := ((k - 1)* numVertices) + spot;
          pcou := pcou + 1;
        else
          Neg[ncou,1] := ((k - 1)* numVertices) + spot;
          ncou := ncou + 1;
        fi;
      fi;
    od;
  od;
  top := top + 20;
  if j = (multiples - 1) then bottom := numEdges
  else bottom := bottom + 20;
  fi;
od;
print('Pos is', Pos,'Neg is',Neg);
pcou := pcou - 1;
temp := submatrix(Pos,1..(pcou),1..1);
Pos := copy(temp);
ncou := ncou - 1;
temp := submatrix(Neg,1..ncou,1..1);
Neg := copy(temp);
qsort(Pos,1,pcou,1);
qsort(Neg,1,ncou,1);
writedata(pos.ione,Pos,integer);
writedata(neg.ione,Neg,integer);

```

```

    print('Dimensions of d',ione, 'are',numEdges,'by',numVertices);
end;

```

*The function **kneser** returns a list of all the edges in the Kneser graph,  $G(n, r, q)$ . The function returns a list of lists type.*

***n**, **r** are natural numbers which are represented in  $G(n, r, q)$ .*

***q** is a nonnegative number which is represented in  $G(n, r, q)$ .*

```

kneser := proc(n,r,q)
    local Points, temp, numberOfPoints,i,j,t,Edges,ipoint,jpoint,common,
        size,flag,A,B,C;
    Points := choose(n,r);
    numberOfPoints := nops(Points);
    B := matrix(1,2,0);
    Edges := matrix(1,2,0);
    flag := 0;
    writeto(vert.n.r.q);
    print('knland7.ms: The verticies are: ',Points);
    writeto(terminal);
    for i to (numberOfPoints - 1) do
        A := matrix(numberOfPoints - i, 2, 0);
        t := 1;
        ipoint := convert(Points[i],set);
        for j from i + 1 to numberOfPoints do
            jpoint := convert(Points[j],set);
            common := ipoint intersect jpoint;
            size := nops(common);
            if size = q then
                A[t,1] := i;
                A[t,2] := j;
                t := t+1;
                flag := flag + 1;
            fi;
        od; if (1 < t) and (t < numberOfPoints - i + 1) then
            C := delrows(A,t..numberOfPoints - i);
            Edges := stack(C,B);
        else

```

```

        if flag > 0 and t > 1 then
            Edges := stack(A,B);
        else Edges := B;
        fi;
    fi;
    if t = 1 and flag = 0 then
        print('There are no edges from the vertex', ipoint,
            'to the other vertices in the graph');
    fi;
    temp := rowdim(Edges);
    if (i = 1) and (t <> 1) then
        A := delrows(Edges,temp..temp);
        Edges := copy(A);
    fi;
    B := copy(Edges);
od;
temp := convert(Edges,listlist);
Edges := convert(temp,list);
RETURN(eval(Edges));
end;

```

*The purpose of NOEDGES is to help determine if the neighbourhood complex will be empty or not empty. If it is empty then the computation of the homology is trivial, otherwise it will take a "little" more work. The function NOEDGES returns a boolean value.*

**TRUE** *if there are absolutely no edges between any of the vertices*

**FALSE** *if there is at least one edge in the graph.*

**n, r** *are natural numbers which are represented in  $G(n, r, q)$ .*

**q** *is a nonnegative number which is represented in  $G(n, r, q)$ .*

```

NOEDGES := proc(n,r,q)
    local Points, temp, numberOfPoints,i,j,t,ipoint,jpoint, common,size,Edges;
    Points := choose(n,r);
    numberOfPoints := nops(Points);
    t := 1;
    for i to (numberOfPoints -1) do

```

```

    ipoint := convert(Points[i],set);
    for j from i +1 to numberOfPoints do
        jpoint := convert(Points[j],set);
        common := ipoint intersect jpoint;
        size := nops(common);
        if size = q then
            t := 2;
            break;
        fi;
    od;
od;
RETURN(evalb(t= 1));
end;

```

*The function **knngbdsimplex** returns a list of lists which is a list of all neighbourhood maximal simplexes of the given graph.*

***MaximalSimplexes** is a list of all the edges in the graph, its of type list of lists.*

***n**, **r** are natural numbers which are represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .*

***q** is a nonnegative number which is represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{q})$ .*

```

knngbdsimplex := proc(MaximalSimplexes,n,r,q)
    local amount, counter, t,i,j,u,v,vertices,temp,size,big,l,A,B,C,k, D;
    amount := vectdim(MaximalSimplexes);
    big := 0;
    for i to amount do
        u := MaximalSimplexes[i];
        v := convert(u,list);
        size := vectdim(v);
        big := big + size;
    od;
    temp := matrix(big,1,0);
    counter := 0;
    for i to amount do
        u := MaximalSimplexes[i];
        v := convert(u,list);
        size := vectdim(v);

```

```

        for j to size do temp[counter + j,1] := v[j]; od;
        counter := counter + size;
    od;
    vertices := repetition(temp);
    counter := rowdim(vertices);
    qsort(vertices,1,counter,1);
    temp := array(1..counter);
    for i to counter do
        t := 0;
        A := array(1..big);
        for j to amount do
            u := MaximalSimplexes[j];
            v := convert(u,vector);
            u := convert(v,matrix);
            size := rowdim(u);
            for k to size do
                if vertices[i,1] = v[k] then
                    C := delrows(u,k..k);
                    for l to (size - 1) do A[l+t] := C[l,1]; od;
                    t := t + size - 1;
                    break;
                fi;
            od;
        od;
        B := matrix(t,1);
        for l to t do B[l,1] := A[l]; od;
        A := repetition(B);
        B := transpose(A);
        C := convert(B,set);
        D := convert(C,list);
        temp[i] := D;
    od;
    appendto(kn.n.r.q);
    print('Neighbourhood Maximal Simplexes',temp);
    RETURN(eval(temp));
end;
```

*The procedure knhomology creates the differential files of the complex. It also*

creates other files using **knCI** and **write20** procedures. The purpose of this procedure is to calculate the differential files ("matrices") and thus help determine the homology of the complex.

**Simplexes** is of type array, and represents the elements which are the maximal simplexes of the complex to be calculated

**n**, **r** are natural numbers which are represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{inter})$ .

**inter** is a nonnegative number which is represented in  $G(\mathbf{n}, \mathbf{r}, \mathbf{inter})$ .

```
knhomology := proc(Simplexes,n,r,inter)
  local noMaxSimplexes,LargestSimplex,a,A,size,maxsimp,b,B,c,C,t,q,
  vertices,vertex,M,rowMi2,colMi2,rowKi2,colKi2,rowMi1,colMi1,
  rowKi1,colKi1,Mi2flag,Ki2flag,temp,K,Ksize,Msize,Mi1flag,Ki1flag;
  vertices := numbcmb(n,r);
  noMaxSimplexes := vectdim(Simplexes);
  LargestSimplex := 1;
  for a to noMaxSimplexes do
    A := convert(Simplexes[a],list);
    size := vectdim(A);
    if LargestSimplex < size then LargestSimplex := size fi;
  od;
  vertex := 1;
  M := star(1,Simplexes,noMaxSimplexes);
  writedata(M.n.r.inter,M,integer);
  temp := LargestSimplex - 1;
  write20(M,Mi,temp,n,r,inter);
  Msize := vectdim(M);
  rowMi2 := Msize;
  colMi2 := LargestSimplex;
  Mi2flag := createflag(rowMi2,LargestSimplex-1,Mi,vertices,n,r,inter);
  temp := knCI(Simplexes,LargestSimplex-1,false,Mi2flag,Msize,vertices,n,r,inter);
  rowKi2:= temp[1];
  colKi2:= temp[2];
  b := LargestSimplex-1;
  temp := readdata(all.Ki.n.r.inter.b,integer,LargestSimplex);
  K := convert(temp,listlist);
  writedata(K.n.r.inter,K,integer);
  temp := 'temp';
  Ksize := vectdim(K);
```

```

for b from LargestSimplex by -1 to 2 do
  appendto(kn.n.r.inter);
  temp := knCI(M,b-2,true,[],0,vertices,n,r,inter);
  rowMil := temp[1];
  colMil := temp[2];
  Milflag := createflag(rowMil,b-2,Mi,vertices,n,r,inter);
  temp := knCI(K,b-2,false,Milflag,rowMil,vertices,n,r,inter);
  rowKil := temp[1];
  colKil := temp[2];
  if rowKil > 0 then
    Kilflag := createflag(rowKil,b-2,Ki,vertices,n,r,inter);
    pn(rowKi2,rowKil,Kilflag,b-1,n,r,inter,vertices);
    differential(rowKi2,rowKil,b-1);
  else
    temp := b-1;
    writeto(differential.temp);
    lprint(0);
    lprint(0);
    writeto(terminal);
    break;
  fi;
  appendto(kn.n.r.q);
  rowMi2 := rowMil;
  colMi2 := colMil;
  rowKi2 := rowKil;
  colKi2 := colKil;
  Kilflag:= [];
  Milflag:= [];
  gc();
od;
end;

```

*The purpose of `knngbdcomplex` is to organize some of the procedures. The procedure `knngbdcomplex` “acts” like a secondary politician or administrator. It just organizes some of the procedures. First it will calculate the neighbourhood maximal simplexes by calling the procedure `knngbdsimplex` to do the work and then calculate the differential matrices of the complex by calling the procedure `knhomology`.*

*edges* is a list which represent the edges in the graph.



$n, r$  are natural numbers which are represented in  $G(n, r, q)$ .

$q$  is a nonnegative number which is represented in  $G(n, r, q)$ .

```
knngbdcomplex := proc(edges,n,r,q)
  local A,B,C;
  A := copy(edges);
  B := knngbdsimplex(A,n,r,q);
  A := [];
  knhomology(B,n,r,q);
end;
```

*The purpose of **rknh** is just a safety precaution, to make sure there are edges in the graph in question. The procedure **rknh** “acts” like a primary politician or administrator, it will first check to make sure there are edges in the graph before computing the neighbourhood simplexes. If there are then it proceeds to calculate the differential files by calling **knngbdcomplex**. Note that if the neighbourhood complex is just a set of points the program may crash.*

$n, r$  are natural numbers which are represented in  $G(n, r, q)$ .

$q$  is a nonnegative number which is represented in  $G(n, r, q)$ .

```
rknh := proc(n,r,q)
  local kn,p;
  if NOEDGES(n,r,q) then
    writeto(kn.n.r.q);
    kn := kneser(n,r,q);
    writeto(terminal);
  else
    kn := kneser(n,r,q);
    writeto(kn.n.r.q);
    print('The edges in the kneser graph G',n,r,q,'are',kn);
    writeto(terminal);
    knngbdcomplex(kn,n,r,q);
  fi;
  writeto(terminal);
end;
```

## B.1 Examples for the procedures of the Relative Homology program

### B.1.1 binsearch

$$\text{Let } A = \begin{bmatrix} 7 & 1 \\ 9 & 6 \\ 13 & 3 \\ 15 & 7 \\ 24 & 8 \end{bmatrix}.$$

`binsearch(15,A,1,5,-1)` will return the value 7.

`binsearch(7,A,1,5,0)` will return the value  $-1$ .

`binsearch(15,A,1,3,-100)` will only search through the first 3 rows and thus return the value  $-1$ .

`binsearch(13,A,1,5,3)` will return the value  $-1$ .

### B.1.2 repetition

$$\text{Let } M_1 = \begin{bmatrix} 4 & 4 & 6 \\ 5 & 5 & 5 \\ 1 & 6 & 3 \\ 5 & 5 & 4 \end{bmatrix} \text{ and } N_1 = \begin{bmatrix} 3 & 8 & 9 \\ 7 & 15 & 1 \\ 1 & 4 & 6 \\ 3 & 8 & 9 \end{bmatrix}$$

Then the commands  $M_2 = \text{repetition}(M_1)$  and  $N_2 = \text{repetition}(N_1)$  will produce the following matrices:

$$M_2 = \begin{bmatrix} 4 & 4 & 6 \\ 5 & 5 & 5 \\ 1 & 6 & 3 \\ 5 & 5 & 4 \end{bmatrix} \text{ and } N_2 = \begin{bmatrix} 3 & 8 & 9 \\ 7 & 15 & 1 \\ 1 & 4 & 6 \end{bmatrix}$$

Note that the order of the rows may vary.

### B.1.3 qsort

$$\text{Let } M = \begin{bmatrix} 6 & 3 & 4 \\ 1 & 2 & 6 \\ 4 & 8 & 3 \\ 2 & 5 & 9 \end{bmatrix}$$

`qsort(M,1,4,1)`  $M$  becomes

$$\begin{bmatrix} 1 & 2 & 6 \\ 2 & 5 & 9 \\ 4 & 8 & 3 \\ 6 & 3 & 4 \end{bmatrix}$$

`qsort(M,2,3,1)`  $M$  becomes

$$\begin{bmatrix} 6 & 3 & 4 \\ 1 & 2 & 6 \\ 4 & 8 & 3 \\ 2 & 5 & 9 \end{bmatrix}$$

`qsort(M,1,4,2)`  $M$  becomes

$$\begin{bmatrix} 1 & 2 & 6 \\ 6 & 3 & 4 \\ 2 & 5 & 9 \\ 4 & 8 & 3 \end{bmatrix}$$

`qsort(M,1,3,3)`  $M$  becomes

$$\begin{bmatrix} 4 & 8 & 3 \\ 6 & 3 & 4 \\ 1 & 2 & 6 \\ 2 & 5 & 9 \end{bmatrix}$$

`qsort(M,2,2,2)`  $M$  becomes

$$\begin{bmatrix} 6 & 3 & 4 \\ 1 & 2 & 6 \\ 4 & 8 & 3 \\ 2 & 5 & 9 \end{bmatrix}$$

`qsort(M,1,1,1)` you should obtain an error message

`qsort(M,4,4,1)` you should obtain an error message

**B.1.4 star**

Let  $L = [[1, 2, 4], [6, 4, 1], [2, 3], [1, 7], [b, c, d, e]]$ .

$\text{star}(1, L, 5)$  will return  $[[1, 2, 4], [6, 4, 1], [1, 7]]$ .

$\text{star}(6, L, 5)$  will return  $[[6, 4, 1]]$ .

$\text{star}(10, L, 5)$  will return  $[]$ .

$\text{star}(4, L, 5)$  will return  $[[1, 2, 4], [6, 4, 1]]$ .

$\text{star}(1, L, 3)$  will return  $[[1, 2, 4], [6, 4, 1]]$ .

$\text{star}(a, L, 5)$  will return  $[]$ .

$\text{star}(d, L, 5)$  will return  $[[b, c, d, e]]$ .

**B.1.5 hash**

Let  $A = [1, 4, 6]$ .

$\text{hash}(A, 3, 6)$  will return 9.

$\text{hash}(A, 3, 7)$  will return 11.

$\text{hash}(A, 1, 6)$  will return 1.

$\text{hash}(A, 3, 4)$  will result in an error.

$\text{hash}([1, 2, 3, 4], 1, 5)$  will return 1.

**B.1.6 createflag**

Examples B.1.1  $\text{createflag}(3, 2, \text{Mi}, 10, 5, 2, 0)$

$$M_2 = \begin{bmatrix} 1 & 4 & 7 \\ 1 & 3 & 6 \\ 1 & 2 & 5 \end{bmatrix} \text{ and } M_2 \text{flag} = \begin{bmatrix} 3 & 3 \\ 11 & 2 \\ 18 & 1 \end{bmatrix}$$

*createflag(12,11,Mi,28,8,2,1)*

$M_{11},$												$M_{11}flag$	
1	2	3	4	5	6	13	18	22	25	27	28	69340	1
1	2	3	5	6	7	10	15	19	23	24	25	120367	12
1	2	9	10	11	12	13	14	15	16	17	18	286171	6
1	3	8	10	11	12	13	14	19	20	21	22	769739	2
1	3	4	5	6	7	8	14	15	16	17	18	5324237	5
1	2	3	4	6	7	11	16	20	23	26	27	8304153	4
1	4	8	9	11	12	13	15	19	23	24	25	2067896	9
1	7	8	9	10	11	12	18	22	25	27	28	5126980	3
1	2	4	5	6	7	9	14	19	20	21	22	10222430	7
1	5	8	9	10	12	13	16	20	23	26	27	11347606	10
1	6	8	9	10	11	13	17	21	24	26	28	11986616	11
1	2	3	4	5	7	12	17	21	24	26	28	12336563	8

*createflag(21,1,Ki,10,5,2,0)*

$K_1 =$	2	8	$K_1flag =$	10	17
	6	10		11	14
	6	7		15	1
	7	10		16	15
	7	9		18	8
	5	7		22	18
	5	9		24	9
	3	4		29	16
	3	10		30	10
	4	10		31	11
	5	6		32	6
	5	8		33	12
	6	8		34	7
	2	4		36	3
	2	9		37	13
	4	9		39	2
	2	3		41	5
	3	8		42	4
	8	10		43	20
	8	9		44	19
	9	10		45	21

**B.1.7 matrixchoose****B.1.8 write20**

**Examples B.1.2** Also note that the values for *name*, *iplace*, *n*, *r*, *q* are purely cosmetic in this procedure. The only purpose they serve is to be consistent with the other procedures. So for this example the values for *name*, *iplace*, *n*, *r*, *q* will be chosen randomly. Let  $A = (a_{i,j})$  where  $a_{i,j} = [i, i + 2]$  and  $1 \leq i \leq 2$ . Let  $B = (b_{i,j})$  where  $b_{i,j} = [i, i + 20]$  and  $1 \leq i \leq 20$ . Let  $C = (c_{i,j})$  where  $c_{i,j} = [i, i + 23]$  and  $1 \leq i \leq 23$ .

`write20(A,first,1,2,3,4)` will generate a file called *first12341* which contains the following data

$$\begin{array}{l} \text{first12341} \\ \left[ \begin{array}{cc} 1 & 3 \\ 2 & 4 \end{array} \right]. \end{array}$$

`write20(B,second,10,20,30,40)` will generate a file called *second102030401* which contains the following data

$$\begin{array}{l} \text{second102030401} \\ \left[ \begin{array}{cc} 1 & 21 \\ 2 & 22 \\ 3 & 23 \\ \vdots & \vdots \\ 20 & 40 \end{array} \right]. \end{array}$$

`write20(C,third,4,3,2,1)` will generate the files *third43211* and *third43212* which will contain the following data

$$\begin{array}{l} \text{third43211} \\ \left[ \begin{array}{cc} 1 & 24 \\ 2 & 25 \\ 3 & 26 \\ \vdots & \vdots \\ 20 & 43 \end{array} \right] \end{array} \quad \begin{array}{l} \text{third43212} \\ \left[ \begin{array}{cc} 21 & 44 \\ 22 & 45 \\ 23 & 46 \end{array} \right] \end{array}$$

**B.1.9 knCI**

*The Petersen graph*

**(K<sub>2</sub>) knCI(K2,2,false,flag2,3,10,5,2,0)**

where  $K_2 = [[8, 9, 10], [6, 7, 10], [5, 7, 9], [5, 6, 8], [3, 4, 10], [2, 4, 9], [2, 3, 8]]$ ,  
and *flag2* is the flag matrix in B.1.1. This will produce two files called *Ki5201* and *Ki5202* using the procedure *write20*. The actual matrix of  $K_2$  in the Kneser graph  $G(5, 2, 0)$  is

$$\begin{bmatrix} 3 & 4 & 10 \\ 5 & 6 & 8 \\ 2 & 4 & 9 \\ 2 & 3 & 8 \\ 8 & 9 & 10 \\ 6 & 7 & 10 \\ 5 & 7 & 9 \end{bmatrix}$$

**(M<sub>1</sub>) knCI([[1, 4, 7], [1, 3, 6], [1, 2, 5]],1,true,[],10,5,2,0)**

$$\begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 4 & 7 \\ 3 & 6 \\ 2 & 5 \\ 1 & 4 \\ 1 & 7 \\ 1 & 6 \\ 1 & 5 \end{bmatrix}$$

**knCI(M1,1,false,flag1,3,10,5,2,0)**

where  $M_1 = [[8, 9, 10], [6, 7, 10], [5, 7, 9], [5, 6, 8], [3, 4, 10], [2, 4, 9], [2, 3, 8]]$ ,  
and *flag1* is the flag matrix for  $M_1$  matrix in B.1.9. This will produce  
the files called *Ki5211* and *Ki5212* after the procedure *write20* is called.  
The entire matrix is:





**B.1.11 pn**

**pn(7,21,flag,5,2,0,10)** where *flag* is the flag matrix of  $K_1$  in the Example B.1.1 Considering the matrices in examples B.1.9 and B.1.9 which are the  $K_2$  and  $K_1$  cells of the Kneser graph with  $star(1)$  removed, and determining the positive and negative locations of  $\pm 1$  we get the following files *pos2* and *neg2* which contain the following data.

```

pos2
 8
10
32
34 neg29
56 33
58 57
80 64
81 103
104 107
105 133
108
109
131
132

```

**B.1.12 kneser**

**kneser(5,2,0)** will return  $[[7, 8], [6, 9], [5, 10], [4, 5], [4, 6], [4, 8], [3, 5], [3, 7], [3, 9], [2, 6], [2, 7], [2, 10], [1, 8], [1, 9], [1, 10]]$

**kneser(7,3,1)** will return  $[[30, 31], [30, 32], [30, 35], [29, 31], [29, 33], [29, 35], [28, 32], [28, 33], [28, 35], [27, 28], [27, 31], [27, 34], [27, 35], [26, 29], [26, 32], [26, 34], [26, 35], [25, 30], [25, 33], [25, 34], [25, 35], [24, 26], [24, 27], [24, 28], [24, 29], [24, 31], [24, 32], [24, 35], [23, 25], [23, 27], [23, 28], [23, 30], [23, 31], [23, 33], [23, 35], [22, 25], [22, 26], [22, 29], [22, 30], [22, 32], [22, 33], [22, 35], [21, 22], [21, 25], [21, 26], [21, 29], [21, 30], [21, 31], [21, 34], [21, 35], [20, 23], [20, 25], [20, 27], [20, 28], [20, 30], [20, 32], [20, 34], [20, 35], [19, 24], [19, 26], [19, 27], [19, 28], [19, 29], [19, 33], [19, 34], [19, 35], [18, 19], [18, 21], [18, 23], [18, 25], [18, 27], [18, 29], [18, 31], [18, 33], [18, 34], [17, 20], [17, 21], [17, 24], [17, 26], [17, 27], [17, 30], [17, 31], [17, 32], [17, 34], [16, 22], [16, 23], [16, 24], [16, 28], [16, 29], [16, 30], [16, 31], [16, 32], [16, 33], [15, 18], [15, 20], [15, 21], [15, 22], [15, 23], [15, 26], [15, 27], [15, 28], [15, 29], [15, 31], [15, 32], [15, 35], [14, 17], [14, 19], [14, 21], [14, 22], [14, 24], [14, 25], [14, 27], [14, 28], [14, 30],$

[14, 31], [14, 33], [14, 35], [13, 17], [13, 18], [13, 19], [13, 20], [13, 23], [13, 24], [13, 25], [13, 26], [13, 29], [13, 30], [13, 32], [13, 33], [13, 35], [12, 13], [12, 16], [12, 19], [12, 20], [12, 23], [12, 24], [12, 25], [12, 26], [12, 29], [12, 30], [12, 31], [12, 34], [12, 35], [11, 14], [11, 16], [11, 18], [11, 19], [11, 21], [11, 22], [11, 24], [11, 25], [11, 27], [11, 28], [11, 30], [11, 32], [11, 34], [11, 35], [10, 15], [10, 16], [10, 17], [10, 20], [10, 21], [10, 22], [10, 23], [10, 26], [10, 27], [10, 28], [10, 29], [10, 33], [10, 34], [10, 35], [9, 10], [9, 11], [9, 13], [9, 16], [9, 17], [9, 18], [9, 21], [9, 23], [9, 24], [9, 25], [9, 26], [9, 28], [9, 32], [9, 33], [9, 34], [8, 12], [8, 14], [8, 15], [8, 19], [8, 20], [8, 22], [8, 25], [8, 26], [8, 28], [8, 32], [8, 33], [8, 34], [7, 8], [7, 10], [7, 12], [7, 14], [7, 16], [7, 17], [7, 20], [7, 22], [7, 24], [7, 25], [7, 27], [7, 29], [7, 31], [7, 33], [7, 34], [6, 8], [6, 11], [6, 12], [6, 15], [6, 16], [6, 18], [6, 19], [6, 22], [6, 23], [6, 26], [6, 27], [6, 30], [6, 31], [6, 32], [6, 34], [5, 8], [5, 13], [5, 14], [5, 15], [5, 17], [5, 18], [5, 19], [5, 20], [5, 21], [5, 28], [5, 29], [5, 30], [5, 31], [5, 32], [5, 33], [4, 5], [4, 6], [4, 7], [4, 10], [4, 11], [4, 13], [4, 16], [4, 17], [4, 18], [4, 19], [4, 20], [4, 22], [4, 27], [4, 29], [4, 30], [4, 32], [4, 33], [4, 34], [3, 5], [3, 6], [3, 8], [3, 9], [3, 10], [3, 12], [3, 14], [3, 16], [3, 17], [3, 19], [3, 21], [3, 23], [3, 26], [3, 28], [3, 30], [3, 31], [3, 33], [3, 34], [2, 5], [2, 7], [2, 8], [2, 9], [2, 11], [2, 12], [2, 15], [2, 16], [2, 18], [2, 20], [2, 21], [2, 24], [2, 25], [2, 28], [2, 29], [2, 31], [2, 32], [2, 34], [1, 6], [1, 7], [1, 8], [1, 9], [1, 13], [1, 14], [1, 15], [1, 17], [1, 18], [1, 22], [1, 23], [1, 24], [1, 25], [1, 26], [1, 27], [1, 31], [1, 32], [1, 33]]

### B.1.13 NOEDGES

**NOEDGES(5,2,0)** *will return false*

**NOEDGES(7,3,1)** *will return false*

**NOEDGES(5,3,0)** *will return true*

### B.1.14 knngbdsimplex

**G(5,2,0)**

**G(7,3,1)**

### B.1.15 knhomology

**knhomology**([[8, 9, 10 , [6, 7, 10], [5, 7, 9], [5, 6, 8], [3, 4, 10], [2, 4, 9], [2, 3, 8], [1, 4, 7], [1, 3, 6], [1, 2, 5]],5,2,0)] *Without actually showing you all the files and data collected I will merely list the information given to a certain extent. First it will determine the star of vertex 1, which is actually found in Example B.1.1 . Second, it determines  $M_2$  which is star of vertex 1. Third, it calculates  $K_2$  which is found in Example B.1.9.*

*Fourth, it will determine  $M_1$  which is found in Example B.1.9. Fifth, it will determine  $K_1$ , which is found in Example B.1.9. Sixth, it will now create the files pos2 and neg2 and then differential2 which are found in Examples B.1.11 and B.1.10 respectively. Seventh, it will calculate  $M_0$  and  $K_0$  appropriately and then calculate differential1 file using the pos1 and neg1 files. The procedure then stops here, and the calculation of the  $D_0$  matrix is left to the user. Please note that knhomology uses other procedures to do the calculations and the examples of their use is found in the appropriate sections.*

## Appendix C

### Ismith Program

To compute integral Smith normal forms we used a new space efficient deterministic algorithm provided by Storjohann [Sto97a]. The new algorithm combines ideas from [Sto96b, Sto96a, Sto97b] and [Abd97] and is based on the *black-box* model of computation, that is, the input matrix is used only to compute matrix-vector products; this is especially suited to the large sparse input matrices arising in our work. For example, let  $A$  be an  $n \times m$  rank  $r$  input matrix with small (one or two decimal digit) entries. The cost of the black-box algorithm is  $O(r^2)$  integer matrix-vector products plus  $O(r)$  integer polynomial multiplications with degree  $r$  univariate polynomials having integer coefficients bounded in length by  $O(r)$  bits. If we assume  $A$  is sparse, with on the order of  $O(n \log n)$  nonzero entries, and we perform intermediate integer operations in a residue number system, then the cost of the algorithm is about  $O(nmr^2)$  bit operations using standard (quadratic) integer arithmetic (ignoring logarithmic terms). This complexity result matches that of the fastest algorithm for dense matrices [Sto96b]. The crucial practical advantage of the black-box algorithm is an improved space complexity; only  $O(r^2)$  additional bits of storage space are required as opposed to  $O(r^3)$  for the previously fastest dense algorithm. The improved space complexity makes the new algorithm practical for the large sparse input matrices we have encountered in our work.

Giesbrecht [Gie96] has proposed a significantly faster algorithm for sparse matrices, also based on the black-box model of computation, that requires only  $O(r)$

matrix-vector products. The drawback of Giesbrecht's algorithm is that it requires randomization and may return, with an exponentially small probability of error, an incorrect integral Smith normal form. The algorithm we have used here, although requiring a factor of  $O(r)$  more matrix-vector products, is deterministic and guarantees correctness of the output. A significant open problem is to find an algorithm which requires only  $O(r)$  matrix-vector products *and* guarantees correctness of the output.

## Bibliography

- [Abd97] Jounaidi Abdeljaoued. *Algorithmes Rapides pour le Calcul du Polynôme Caractéristique*. PhD thesis, l'Université de Franche-Comté, March 1997.
- [Gie96] Mark Giesbrecht. Probabilistic computation of the Smith normal form of a sparse integer matrix. In H. Cohen, editor, *Algorithmic Number Theory: Second International Symposium*, pages 175–188, 1996. Proceedings to appear in Springer's Lecture Notes in Computer Science.
- [Sto96a] Arne Storjohann. Computing Hermite and Smith normal forms of triangular integer matrices. Technical Report 256, Departement Informatik, ETH Zürich, December 1996.
- [Sto96b] Arne Storjohann. A fast+practical+deterministic algorithm for triangularizing integer matrices. Technical Report 255, Departement Informatik, ETH Zürich, December 1996.
- [Sto97a] Arne Storjohann, 1997. Personal Communication.
- [Sto97b] Arne Storjohann. A solutions to the extended gcd problem with applications. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '97*, pages 109—116, 1997.

# Appendix D






## Notation

$\pm$	plus or minus
$\subseteq$	subset
$\subset$	proper subset
$\Rightarrow$	implies
$\Leftrightarrow$	if and only if
$\square$	Q.E.D.
$\in$	member of
$\cap$	intersection
$\cup$	union
$\oplus$	direct sum
$A \equiv B$	$B$ is the integral Smith normal form of $A$
$\approx$	isomorphic
$\sim$	homologous
$\simeq$	homotopic
$\cong$	homeomorphic
$\hookrightarrow$	inclusion map
$\twoheadrightarrow$	epic (onto) map
$\dashv$	functor
$a b$	$a$ divides $b$

$\mathbb{Z}$	the set of integers
$\mathbb{R}$	the set of reals
$\mathbb{R}^n$	the Euclidean space of dimension $n$
$\mathbb{Z}_m$	cyclic group of order $m$
$I_m$	$m \times m$ identity matrix
$0_{m \times n}$	$m \times n$ zero matrix
$M_{m,n}(R)$	set of all $m \times n$ matrices over $R$
$S^{(\tau)}$	$\tau$ -skeleton of a simplicial complex $S$
$S^n$	$n$ -sphere
$\emptyset$	empty set
$\text{im } f$	image of the map $f$
$\text{ker } f$	kernel of the map $f$
$\pi$	projection map
$i$	inclusion map
$[z]$	the homology class of $z$ i.e. $[z] = z_+ B_n$
$[f]$	equivalence class of $f$ w.r.t. homotopy
$ A $	the cardinality of $A$ , where $A$ is a set
$ K $	polyhedron of a simplicial complex $K$



$I$	unit interval, $I = [0, 1]$
$1_X$	the identity function of an object $X$
$\mathbf{P}(V)$	the power set of $V$
$\mathcal{K}^a$	the set of all abstract simplicial complexes
$\mathcal{K}$	the set of all geometrical simplicial complexes
$U$	underlying abstract simplicial complex map
$\{v_1, \dots, v_n\}$	$n$ -simplex with vertices $v_1, \dots, v_n$
$d_n$	$n$ th differential map
$D_n$	$n$ th differential matrix for $d_n$
$H_n(X)$	$n$ th homology group of $X$
$Z_n(X)$	$n$ -cycles of $X$
$B_n(X)$	$n$ -boundaries of $X$
$H_n(X, Y)$	$n$ th relative homology pair
$C_n(X)$	$n$ -chains of an abstract simplicial complex $K$
$G(n, r, q)$	generalized Kneser graph
$\text{diag}(x_1, \dots, x_n)$	is an $n \times n$ matrix with $x_1, \dots, x_n$ along the main diagonal and zero entries elsewhere.

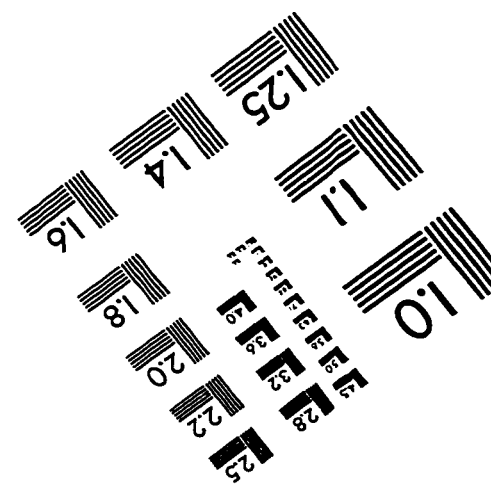
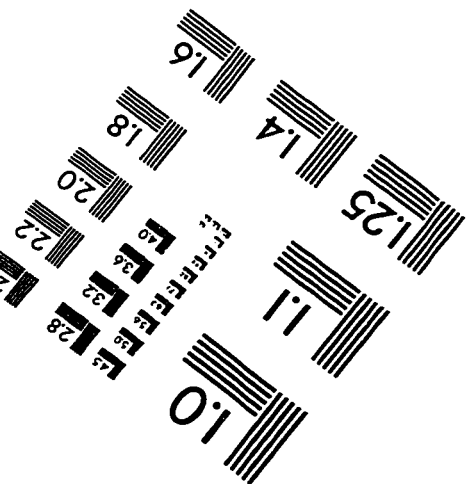
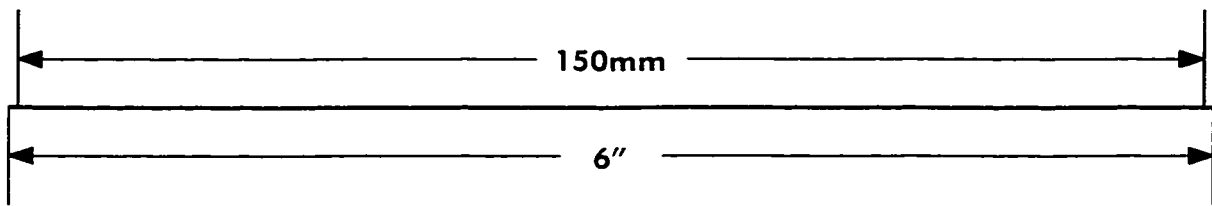
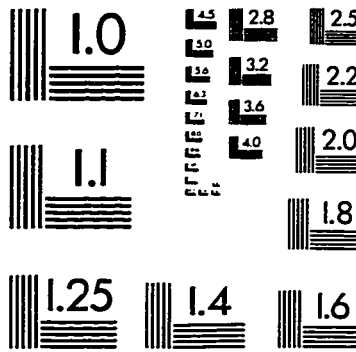
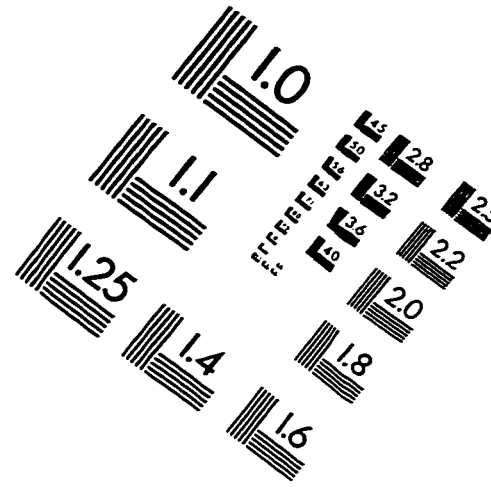
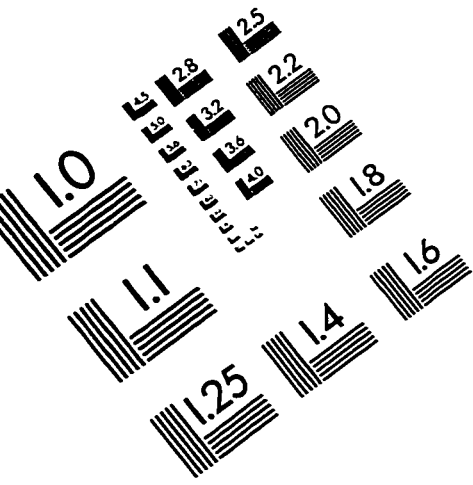
	hollow
	two-dimensionally filled
	three-dimensionally filled
	vertex
	edge

## Bibliography

- [1] Noga Alon, Péter Frankl, László Lovász, The Chromatic Number of Kneser Hypergraphs, Transactions American Mathematical Society, Volume 298, Number 1, (1986), p.359-370.
- [2] Norman L. Biggs, E. Keith Lloyd, Robin J. Wilson, *Graph Theory 1736-1936*, Clarendon Press, Oxford (1976).
- [3] Eduard Čech, Höherdimensionalen Homotopiegruppen, Verhandl. des Inter. Math. Kongresses, Zürich, 1932, Bd.2, p.203.
- [4] Keith Devlin, *Mathematics: The New Golden Age*, Penguin Books, New York (1988).
- [5] Peter John Giblin, *Graphs, Surfaces, and Homology: an Introduction to Algebraic Topology*, Chapman and Hall, London (1977).
- [6] Brian Hartley, Trevor O. Hawkes, *Rings, Modules and Linear Algebra*, Chapman and Hall Mathematics Series (1970), London, 2nd printing (1995).
- [7] László Lovász, Kneser's Conjecture, Chromatic Number and Homotopy, Journal of Combinatorial Theory Series A, 25, (1978), p.319-324.
- [8] R. James Milgram, Peter Zvengrowski, An Application of Principal Bundles to Coloring of Graphs and Hypergraphs, Supp. ai Rend. del Circ. Mat. di Palermo Serie II, Num.37, (1994), p.161-167.
- [9] R. James Milgram, Peter Zvengrowski, On a Theorem of Lovász, Preprint.

- [10] Henri Poincaré, *Analysis Situs*, Comptes-Rendus Note, (1895), p.193-288.
- [11] Joseph J. Rotman, *An Introduction to Algebraic Topology*, Springer, Graduate Texts in Mathematics 119, New York (1988).
- [12] Edwin H. Spanier, *Algebraic Topology*, McGraw-Hill, New York, (1966).
- [13] Arne Storjohann, A fast+practical+deterministic algorithm for triangularizing integer matrices, Technical Report 255, Departement Informatik, ETH Zürich, (December 1996).
- [14] Douglas B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, N.J. (1996).

# IMAGE EVALUATION TEST TARGET (QA-3)



**APPLIED IMAGE, Inc**  
1653 East Main Street  
Rochester, NY 14609 USA  
Phone: 716/482-0300  
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved