A Lower Bound on the Size of the Smallest Kochen-Specker Vector System in Three Dimensions

Felix Arends (arends@maths.ox.ac.uk)

Candiate Number: 633146

Mathematical Institute, University of Oxford

Supervisor: Dr Joël Ouaknine

Submitted in partial fulfilment of the requirements for the

Degree of Master of Science

in

Mathematics and the Foundations of Computer Science

at the

University of Oxford

Abstract

Kochen-Specker (KS) vector systems in three dimensions are sets of directions in $\mathbb{R}^3$ with the property that it is impossible to assign 0s and 1s to the directions in such a way that no two orthogonal directions are assigned 0 and no three mutually orthogonal directions are assigned 1. The existence of such sets forms the basis of the proofs of the Free Will Theorem and the Kochen-Specker Theorem. Currently the smallest known KS vector system in three dimensions contains 31 directions. We give an exhaustive algorithm that can be used to show that the smallest such set must contain at least 18 directions. We show how to represent KS vector systems by graphs that are not '101-colourable'. The NP-completeness of checking 101-colourability is shown and several necessary properties of the graph representation of the smallest KS vector system are found.

Acknowledgements

I thank Dr Joël Ouaknine for suggesting this interesting topic for my dissertation and for being a great supervisor; for the inspiring discussions in his office, for providing me with many excellent resources and ideas when I started my work, and for helping me get in touch with experts in various fields when I needed support.

There are several more people whom I would like to thank for their help and their contributions: Prof. Charles W. Wampler, who explained to me very patiently the theory behind the Bertini software package and who helped me produce some Bertini input files that were central to my work; Dr Jean-Pierre Merlet for sharing some of his experience with finding Kochen-Specker vector systems using interval analysis; Prof. D. Knuth for pointing me to a most interesting paper on orderly graph generation, which provided a valuable addition to my thesis; Prof. Nick Trefethen for establishing the contact with Charles Wampler.

Finally, I would like to thank the Computing Laboratory, the Mathematical Institute, and St Cross College for financially supporting my attending of the Summer School on Mathematical Methods in Computational Kinematics in Innsbruck.

Contents

Chapter 1

Introduction

1.1 Motivation

The topic of this dissertation was suggested by my supervisor, Dr Joël Ouaknine. He had attended a talk given by Prof. John H. Conway about the "Free Will Theorem" [4]. In his talk, Conway formulated the problem that is the topic of my dissertation as an open problem. I had attended the same talk at a different location, so I could immediately relate to the problem.

The statement of the Free Will Theorem can be roughly explained as follows. There are certain 'spin-1 particles' (e.g. photons), whose 'spin' (a physical property) can be measured along any given direction. The squared outcome of such a measurement is either 0 or 1. The theorem says that if experimenters can choose the direction from which to perform a spin 1 experiment freely (i.e. not determined by the past), then the response of a spin 1 particle to such an experiment is also not determined by the past:

Theorem 1 (The Free Will Theorem)**.** *If the choice of directions in which to perform spin 1 experiments is not a function of the information accessible to the experimenters, then the responses of the particles are equally not functions of the information accessible to them.*

The theorem assumes three axioms, called the SPIN, the TWIN, and the FIN axiom. A more recent, stronger version of the theorem replaces the FIN axiom with a weaker MIN

axiom [5].

The axiom that will be of interest to us is the SPIN axiom:

Axiom 1 (The SPIN Axiom)**.** *Measurements of the squared (components of) spin of a spin 1 particle in three orthogonal directions always give answers 1, 0, 1 in some order.*

This axiom not only follows from the postulates of quantum mechanics, but it can and has also been verified experimentally [12]. This axiom alone already gives rise to an interesting 'paradox', called the Kochen-Specker Paradox [5]. If the response of a particle to any spin measurement was determined prior to the actual measurement, then those responses would define a function from the three-dimensional unit sphere to the set $\{0, 1\}$ with the so-called 101 property: any three points on the unit sphere with mutually orthogonal position vectors must be assigned the values 1, 0, 1 in some order. It is a remarkable fact that such a function does not exist.

The non-existence of such a '101-function' can be proved by exhibiting a finite set of points on which such a function cannot be defined. In the proof of the Free Will Theorem, Conway and Kochen present a set of 33 points. A brief argument suffices to show that no 101-function can be defined on those 33 points. Conway also found a set of 31 points which does not admit a 101-function (e.g. p. 114 of [19]). The question he phrased as an open problem during his talk, and the topic of this dissertation, is what the smallest possible such set is. A considerable amount of work has already been put into finding such small sets: the first known set contained 117 points [13]. This number was subsequently improved to 33 and then to 31 (cf. pp. 2-3 of [16]).

Finding a smaller Kochen-Specker vector systems is desirable, because it would 'simplify' the proofs of the Free Will Theorem and the Kochen-Specker Theorem in some sense. Of course, the proof that no 101-function can be defined on a set of directions may be more involved for a small set than for a large set. A small set of directions would also simplify the execution of the experiment described in the proof of the Free Will Theorem.

1.2 Outline

The main body of this dissertation consists of four parts. First, the notion of a Kochen-Specker vector system – a set of directions that is sufficient to derive the Kochen-Specker Paradox from the SPIN Axiom – is defined. We show how to represent Kochen-Specker vector systems as graphs that cannot be coloured in a certain way. Next, we deal with the problem of finding a set of directions whose graph representation is a given graph, i.e., embedding a graph into the sphere (or more accurately the projective plane). The last two sections deal with establishing a lower bound and an upper bound on the size of the smallest Kochen-Specker vector system in three dimensions.

We will derive a number of conditions that the smallest Kochen-Specker vector system must satisfy. Unfortunately, the number of graphs satisfying those conditions is still very large.

1.3 Notation

Most of this work is concerned with simple, undirected graphs. Often, we will refer to simple, undirected graphs simply as "graphs". We will define the notion of "101-colourability" of a graph. Sometimes we will call a graph "uncolourable", by which we mean that it is not 101-colourable. The 4 cycle C_4 is called a *square*. A simple, undirected graph is called square-free iff it contains no subgraph isomorphic to C_4.

Three-dimensional vectors are denoted by bold letters, such as $\mathbf{x}$ and $\mathbf{y}$. The expression $\mathbf{x} \cdot \mathbf{y}$ denotes the scalar product between vectors $\mathbf{x}$ and $\mathbf{y}$. The x, y, and z components of a vector $\mathbf{x}$ are denoted $x(1)$, $x(2)$, and $x(3)$, respectively. In three-dimensional figures and illustrations, the vertical axis is taken to be the y axis; the x axis runs left to right and the positive z axis runs "into the paper".

In this work, we clearly distinguish between three kinds of proofs. The usual kind of proof is the *mathematical* proof. In addition to mathematical proofs, we also present some results that have *computer-aided* proofs. This means that a computer program was involved in proving the result. Formally, such a proof may not be considered a mathematical proof: it would also have to include proofs of the correctness of the computer program that was used,

and of the correctness of the compiler and computer architecture of the machine on which the computer program was executed. Finally, we will also present some results that have *numerical* proofs: by a numerical proof, we mean a proof for which a computer program was used, and floating point arithmetic was involved in a way that cannot be shown to be entirely accurate. It is reasonable to assume that the computed results are correct up to a certain accuracy, and the initial problem would have to be very ill-conditioned in order for the proof to be invalid. However, such a numerical proof is in some sense only a strong indication of the truth rather than an actual proof. A computer-aided proof is certainly much stronger than a numerical proof.

Chapter 2

Kochen-Specker Vector Systems

We recall the SPIN Axiom from the introduction:

Axiom 2 (The SPIN Axiom)**.** *Measurements of the squared (components of) spin of a spin 1 particle in three orthogonal directions always give answers 1, 0, 1 in some order.*

As S. Kochen and E. Specker showed [13], this axiom gives rise to a 'paradox': There is no function that assigns to each point on the unit sphere either value 0 or value 1 in such a way that every triple of points with mutually orthogonal position vectors is assigned 1, 0, 1 in some order. A finite set of points on the unit sphere suffices to arrive at a contradiction when assuming the existence of such a function. We investigate the lower and the upper bound on the size of the smallest such set.

2.1 Definitions

2.1.1 101-functions

We define the notion of a 101-function ("one-oh-one function") as in [4]. First, observe that if the outcomes of all possible spin measurements of a spin 1 particle were pre-determined, then the TWIN Axiom would imply that we can define a function on the unit sphere of directions with the property that any three mutually orthogonal directions are assigned $1, 0, 1$ in some order. We can easily derive further properties of such a function:

1. Opposite vectors are assigned the same value.

2. No two orthogonal vectors are assigned 0.

3. No three mutually orthogonal vectors are assigned 1.

Note that on a subset of the unit sphere of directions, these three properties are possibly stronger than the requirement that any three mutually orthogonal directions are assigned $1, 0, 1$ in some order (consider for instance a set consisting only of two opposite directions). The three properties motivate the following definition:

Definition 1. *Let $\mathcal{K}$ be a set of directions in $\mathbb{R}^3$. A function $f : \mathcal{K} \to \{0, 1\}$ is called a 101-function iff it satisfies the following three conditions:*

1. *If $u, v \in \mathcal{K}$ and $u \cdot v = 0$, then $f(u) \neq 0$ or $f(v) \neq 0$, i.e. no two orthogonal vectors are assigned value 0 by f.*

2. *If $u, v, w \in \mathcal{K}$ are three mutually orthogonal vectors, then $f(u) \neq 1$, $f(v) \neq 1$ or $f(w) \neq 1$.*

3. *Opposite directions $u, -u \in \mathcal{K}$ satisfy $f(u) = f(-u)$.*

The Kochen-Specker Paradox resulting from the SPIN Axiom is that no 101-function can be defined on $\mathbb{S}^2$.

2.2 Kochen-Specker Vector Systems

Next, we define the notion of a *direction set* in three dimensions:

Definition 2. *A direction set $\mathcal{K} \subseteq \mathbb{R}^3$ is a set of non-zero vectors, none of which are collinear.*

Note that because of the non-collinearity requirement, a function $f : \mathcal{K} \mapsto \{0, 1\}$ is a 101-function on a direction set $\mathcal{K}$ iff it satisfies properties 1 and 2 of Def. 1 (101-function). Finally, we use the following definition of a Kochen-Specker vector system in three dimensions:

Definition 3. *A three-dimensional Kochen-Specker vector system (KS vector system) is a direction set on which no 101-function can be defined.*

2.3 Representing Direction Sets as Graphs

Given a direction set $\mathcal{K}$, we may define a graph $G = (V, E)$ corresponding to this direction set as follows:

$$V = \mathcal{K}$$

$$E = \{(x, y) : x, y \in \mathcal{K} \text{ and } x \cdot y = 0\}$$

In other words, there is one vertex per direction (or, if $\mathcal{K}$ is infinite, there is a bijection between the vertices of G and the elements of $\mathcal{K}$) and two vertices are connected by an edge iff the corresponding directions in $\mathcal{K}$ are orthogonal. Of course, the actual directions in $\mathcal{K}$ are not preserved when constructing G. We can rotate the directions of $\mathcal{K}$ around the origin while the corresponding graph G does not change. However, enough of the structure is preserved to determine whether a 101-function can be defined on $\mathcal{K}$. We call G the *graph representation* of $\mathcal{K}$.

2.3.1 Different Definitions in the Literature

Unfortunately, there is no uniform terminology of terms relating to KS vector systems. The use of the term "Kochen-Specker vector system" was motivated by the title of the research report of M. Pavičić, J-P. Merlet, and N. Megill ([17], "Exhaustive Enumeration of Kochen-Specker Vector Systems"). In this report, the expression "Kochen-Specker set" is defined similarly, but for arbitrary dimensions. Occasionally, the term "Kochen-Specker system" is also used.

However, it must be clarified that the definition of a KS set is somewhat different in the research report. Pavičić et al. require that every pair of vectors in an n-dimensional "Kochen-Specker set" belongs to a set of n mutually orthogonal vectors. Using this definition in three dimensions, one could define a 101-function to be a function that maps any three mutually orthogonal vectors to 1, 0, 1 in some order. The 'stronger' conditions

of our definition would not be stronger on KS sets, and would therefore not be required. This is why, according to their counting, Conway's KS set contains 51 vectors instead of 31. Indeed, the graph representation of Conway's KS system contains 20 edges that do not belong to any triangle: there are 20 pairs of orthogonal vectors that do not belong to a triple of mutually orthogonal vectors. Of course, the 20 additional vectors can easily be added to the set of directions, thus resulting in a set of 51 directions.

This is why care needs to be taken when comparing the result of Pavičić et al. with the results established in this thesis. Their algorithm has enumerated and checked all direction sets of size up to and including 30. However, according to their counting, there are still 21 directions missing until the size of the system suggested by Conway is reached. It is a reasonable assumption that the number of systems that have to be checked increases at least exponentially with the number of directions, although the author is not aware of the exponent's coefficient. Thus, the gap from 30 to 51 directions according to the different counting is probably similar to the gap from 17 to 31 directions according to the counting used in this thesis. The speed of growth of the number of direction sets as defined in this thesis is investigated in the Discussion chapter.

2.3.2 101-colorability of a graph

We define the notion of 101-colourability of a graph. If G is the graph corresponding to a direction set $\mathcal{K}$, then we would like G to be 101-colourable iff there exists a 101 function for $\mathcal{K}$. The definition can be extended to cover all simple undirected graphs, however.

Definition 4. *A simple undirected graph $G = (V, E)$ is 101-colourable iff there exists a function $c : V \mapsto \{0, 1\}$ s.t. not both vertices of any edge are assigned 0 and not all three vertices of any triangle are assigned 1, i.e.*

1. *If $\{a, b\} \in E$ then $c(a) = 1$ or $c(b) = 1$*
2. *If $\{\{x, y\}, \{y, z\}, \{z, x\}\} \subseteq E$ then $c(x) = 0$ or $c(y) = 0$ or $c(z) = 0$*

In order to put 101-colourability in context with regular k-colourability, we show the following result.

Proposition 1. *A 3-colourable graph $G = (V, E)$ is 101-colourable.*

Proof. Let $c : V \mapsto \{1, 2, 3\}$ be a 3-colouring. Define the colouring $c' : V \mapsto \{0, 1\}$ by

$$c'(x) = \begin{cases} 0 & \text{if } c(x) = 1 \\ 1 & \text{otherwise} \end{cases} .$$

We claim that c' is a valid 101-colouring. Indeed, every edge $\{a, b\}$ must have $c'(a) \neq 0$ or $c'(b) \neq 0$, since $c(a) \neq c(b)$. Furthermore, c assigns all three colours 1, 2, and 3 to the three vertices of a triangle. This means that c' assigns 1 to two of the vertices and 0 to one of the vertices, as required. □

NP-completeness

Deciding whether a simple graph admits a 101-colouring is NP-complete. Following the customary naming convention, we introduce the name 101-COLOR for this problem. We show the NP-completeness by showing that the problem 3-COLOR of deciding whether a simple graph is 3-colourable can be reduced to 101-COLOR. The NP-completeness of 101-COLOR then follows from the NP-completeness of 3-COLOR (e.g. [6], pp. 1019f.).

Theorem 2. *The 101-colourability problem is NP-complete.*

Proof. Note that 101-colourability is in NP: there are 2^n possible colourings of a graph on n vertices. Checking whether one such colouring is a valid 101-colouring can be done in polynomial time, e.g. by checking all $\mathcal{O}(n^3)$ triples of vertices whether they contain an 'all 0-edge' or an 'all 1-triangle'.

It remains to show NP-hardness. Given a simple graph $G = (V, E)$, we show how to construct a simple graph $G' = (V', E')$ which is 101-colourable iff G is 3-colorable. Furthermore, it will be clear that the construction can be performed in time proportional to a polynomial of the input size. We assume a "reasonable" encoding of the input instead of over-complicating the proof with many details. Since 3-COLOR is NP-complete, this proves the theorem.

The new graph G' will contain three vertices v_1, v_2, and v_3 for each vertex v in G. Those three vertices are joined to form a triangle. For every edge $e = \{u, v\}$ in G, we introduce

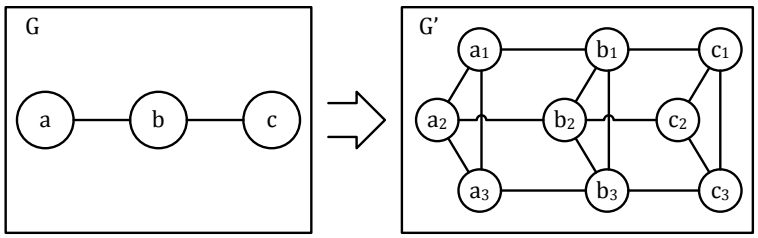

Figure 2.1: An example of the construction of G' from G.

three edges in G': $\{u_1, v_1\}$, $\{u_2, v_2\}$ and $\{u_3, v_3\}$. One can think of G' as three copies of G whereby any vertex of one copy is joined to the two corresponding vertices in the other two copies. This construction is illustrated using a simple example in Fig. 2.1.

We claim that G is 3-colourable iff G' is 101-colorable. This claim is a corollary of the following two lemmas.

Lemma 1. *If G is 3-colourable, then G' is 101-colourable.*

Proof. Suppose G is 3-colourable and let $c : V \mapsto \{1, 2, 3\}$ be a valid 3-colouring. If a vertex $v \in V$ has colour $c(v)$, then we colour the corresponding vertices v_1, v_2, and v_3 by assigning 0 to vertex $v_{c(v)}$ and 1 to the other two vertices. Hence, we encode the 3-colouring of G in our colouring of G'. Formally, our new colouring is defined by $c' : V' \mapsto \{0, 1\}$:

$$c'(v_i) = \begin{cases} 0 & \text{if } i = c(v) \\ 1 & \text{otherwise .} \end{cases} \tag{2.1}$$

We note the following two properties of G' and c':

1. By definition of c', if w_1, w_2 and w_3 are the vertices in G' corresponding to w in G, then exactly one of those will be assigned colour 0.

2. Suppose that u_i and v_j are vertices in G' corresponding to vertices u and v in G, respectively. Suppose further that (u_i, v_j) is an edge in G'. Then it follows from the construction of G' that either $u = v$ and $i \neq j$, or that $u \neq v$ and $i = j$.

It remains to show that c' is indeed a valid 101-colouring of G'. Indeed, neither of the two conditions of Def. 4 (*101-colouring*) is violated:

Suppose, for a contradiction, that the first condition is violated, i.e. that two adjacent vertices u_i and v_j have colour 0. By Property 1 the corresponding vertices u and v of G must be distinct. By Property 2 we must then have $i = j$ and $\{u, v\} \in E$. But by the definition of c', we have $c(u) = c(v)$. This contradicts the assumption that c is a valid 3-colouring of G. We conclude that no two adjacent vertices in G' can have colour 0. The colouring c' therefore satisfies the first condition of Def. 4.

Next, suppose for a contradiction, that the second condition is violated, i.e. some vertices u_i, v_j, and w_k of a triangle in G' have colour 1. Let u, v, and w denote the corresponding vertices in G. We claim that all three of those must be distinct vertices in G. Indeed, by Property 1, not all of u, v, and w can be the same. Wlog. (by symmetry) we may assume $u \neq v$ and therefore (by Property 2) $i = j$. But then $w \neq u$ or else $k \neq i = j$ which implies that $w = v \neq u$. By symmetry we also have $w \neq v$, proving the claim.

Hence, we have that u, v, and w are all distinct and $i = j = k$. By the definition of c' this means that none of u, v, and w can have colour i. Since u_i, v_j, and w_k form a triangle in G', their corresponding vertices form a triangle in G. But then c colours the triangle $\{u, v, w\}$ using only two colours, contradicting the assumed validity of c. The colouring c' therefore also satisfies the second condition of Def. 4 and thus c' is a valid 101-colouring. □

Lemma 2. *If G' is 101-colourable, then G is 3-colourable.*

Proof. Suppose that $c' : V' \mapsto \{0, 1\}$ is a valid 101-colouring of G'. Suppose v_1, v_2, and v_3 are the three vertices in G' corresponding to v in G. Since $\{v_1, v_2, v_3\}$ is a triangle in G', exactly one of those vertices has colour 0. We may therefore define a colouring $c : V \mapsto \{1, 2, 3\}$ of G by $c(v) = i \Leftrightarrow c'(v_i) = 0$.

This is valid 3-colouring: Suppose, for a contradiction, that $k = c(v) = c(u)$ for two adjacent vertices v and u in G. Then $c'(v_k) = c'(u_k) = 0$ in G' and by the construction of G' the two vertices are adjacent. This contradicts the validity of c' (condition 2 of Def. 4). Hence, if G' is 101-colourable, then G is 3-colourable. □

Finally, note that $|V'| = 3\,|V|$ and $|E'| = 3\,|E| + 3\,|V|$, i.e. the size of G' is $\Theta(|E| + |V|)$. Also, the construction of G' from G is straightforward and clearly possible in polynomial time. □

Whenever the 101-colourability of a given graph needed to be checked, the problem was formulated as a SAT instance and solved using the MiniSAT solver [7], which can be linked statically with a C++ program. One variable was introduced for the colour of each vertex. For each edge, a clause stating that not both vertices can have colour 0 was introduced; for each triangle, a clause was added stating that not all three vertices can have colour 1. The MiniSAT solver outperformed a straight-forward backtracking algorithm on all reasonably large graphs.

2.4 Records in three dimensions

In their original paper, Kochen and Specker [13] construct a KS vector system with 117 vectors. Subsequent constructions of smaller KS vector systems are referred to as "records" [17]. We outline the construction of three of those systems.

2.4.1 The Original KS Vector System with 117 Directions

Fig. 2.2 shows the graph from which the original KS Vector System is constructed. The graph contains 120 vertices. To form the graph representation of the KS vector system, three pairs of vertices are identified: A and a, B and b, and C and c. The detailed construction of the direction set which has this graph representation can be found in the original paper [13].

Interestingly, 88 of the 117 directions suffice as a KS vector system (88 directions are also necessary). This number was found using the backtracking algorithm introduced in Sect. 5.2.

2.4.2 Peres' System with 33 Directions

Peres' KS vector system [18] uses only 33 directions to establish a contradiction. It is constructed by taking three copies of the unit cube $[-1, 1]^3$. One copy is rotated 45 degrees around the x-axis, a second cube is rotated 45 degrees around the y-axis, and the last cube is rotated 45 degrees around the z-axis. The directions then correspond to the symmetry axes of the rotated cubes (see Fig. 2.3). The construction is copied from Conway and Specker

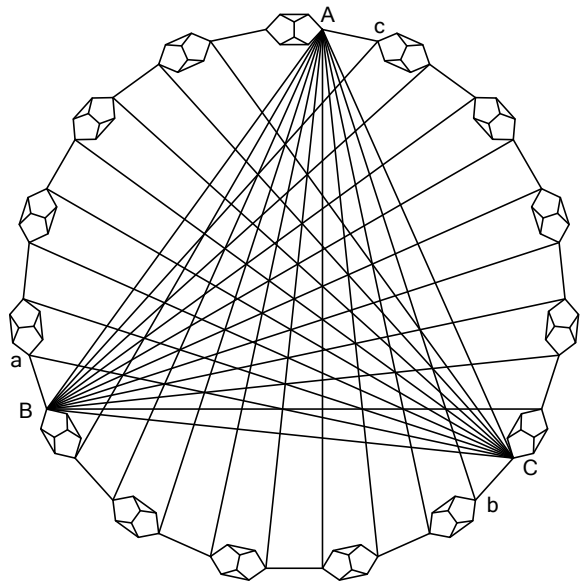

Figure 2.2: The graph from which the original KS vector system was constructed. The pairs of vertices (A, a), (B, b), and (C, c) are identified.

[5]. They also give a concise proof of the fact that no 101-function can be defined on this set of directions.

2.4.3 Conway's System with 31 Directions

The 31 directions of the KS vector system found by John Conway lie on a grid of points on the surface of a cube. The black dots in Fig. 2.4 represent the directions in the set. This figure is taken from [19]. As of the time of this writing, no smaller KS vector system in three dimensions is known.

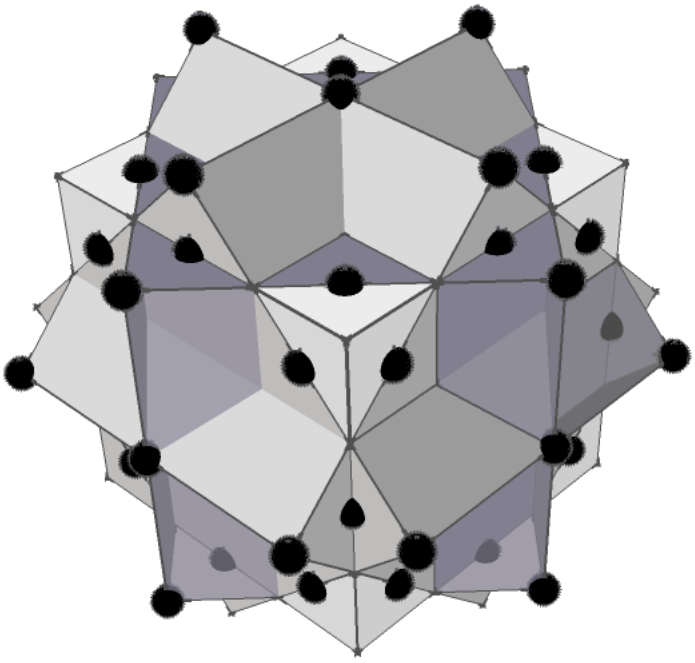

Figure 2.3: Illustration of the directions in Peres' KS vector system. The directions are the symmetry axes of the shaded cubes.

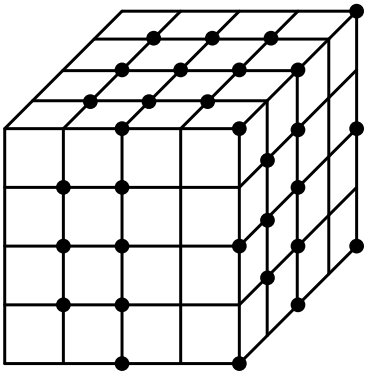

Figure 2.4: A visual representation of the KS vector system discovered by John Conway.

Chapter 3

Directions from Graphs

Unfortunately, not all graphs correspond to a set of directions in three dimensions. For example, the following result states that a graph representing a direction set must not contain a square. We call a graph corresponding to some direction set *embeddable*. All other graphs are called *unembeddable*. Note that every subgraph of an embeddable graph is also embeddable. Equivalently, every graph containing an unembeddable graph is also unembeddable. If a direction set $\mathcal{K}$ has graph representation G, then we call $\mathcal{K}$ an *embedding* of G.

Proposition 2. *A graph representing a direction set must be square-free.*

Proof. It suffices to show that the square (C_4) is not embeddable. It follows immediately

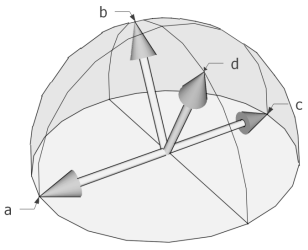

Figure 3.1: A square forces two collinear directions.

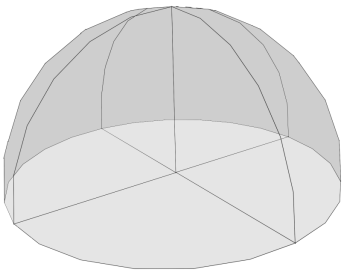

Figure 3.2: The division of the upper half-sphere into four quadrants.

that no graph representing a direction set can contain a square. Note that if a vertex a has two neighbours b and d, then the direction associated with a must be orthogonal to the directions associated with b and d. If we write $\delta(x)$ for the unit vector in the direction associated with vertex x, then $\delta(a) = \pm\delta(b) \times \delta(d)$. This expression is always a valid direction, because $\delta(b)$ and $\delta(d)$ are not allowed to coincide. Suppose now that vertices a, b, c, and d form a square. Then $\delta(a) = \pm\delta(b) \times \delta(d)$ and $\delta(c) = \pm\delta(b) \times \delta(d)$, i.e. $\delta(a) = \pm\delta(c)$. In other words, $\delta(a)$ and $\delta(c)$ are collinear. This violates the condition that all the directions in a direction set must be distinct. □

Our next result states that a finite embeddable graph must be 4-colourable.

Proposition 3. *A finite graph representing a direction set must be 4-colourable.*

Proof. We begin by outlining the idea of the proof and then fill in the details. The direction vectors of a KS vector system can be represented as points of the upper half ($y > 0$) of a unit sphere (we might have to apply a rotation to all the position vectors). This upper half of the unit sphere can be split into four quadrants. Points that are inside each quadrant cannot represent orthogonal directions. Hence, the corresponding vertices in the graph form an independent set. Therefore, the graph consists of four independent sets, which can be considered the colour classes of a 4-colouring.

The quadrants of the upper half-sphere are shown in Figure 3.2. It remains to show that we can always rotate the direction vectors in such a way that none of the points end up on the boundaries of the quadrants.

First, note that there must exist some plane through the origin on which none of the points lie. Indeed, a plane through the origin can be described by its normal vector. Every direction in the KS system lies on exactly those planes with normal vectors orthogonal to the direction. Hence, every direction vector in the KS system forbids a one-dimensional set of normal vectors, which has Lebesgue measure 0 inside the (two-dimensional) set of all normal vectors. Since the KS system is finite by assumption, the union of all 'forbidden' sets also has Lebesgue measure 0. Thus, almost all of the normal vectors describe planes in which no direction vectors of the KS system lie.

Let P be a plane in which none of the direction vectors lie. We rotate the direction vectors around the origin s.t. P is the plane $y = 0$. Next, we rotate around the y axis in such a way that no points lie on the planes described by $x = 0$ and $z = 0$. This can be done for a similar reason: every direction in the KS system excludes (up to) four rotation angles (the angles which cause the x or z coordinates to vanish). Those four angles have Lebesgue measure 0 with respect to the Lebesgue measure defined on $[0, 2\pi)$. Thus, we can rotate around almost all angles without moving any direction onto the $x = 0$ or $z = 0$ plane.

Note that this leaves the possibility of a single point being located at $(x, y, z) = (0, 1, 0)$. But the corresponding direction would not be orthogonal to any other direction, because we have chosen P in such a way that no direction vectors lie in it.

The four quadrants are now defined by $\{x > 0\} \cap \{z > 0\}$, $\{x < 0\} \cap \{z > 0\}$, $\{x > 0\} \cap \{z < 0\}$, and $\{x < 0\} \cap \{z < 0\}$. If a point is located at $(x, y, z) = (0, 1, 0)$ it can be put into either of the four quadrants. The quadrants form the colour classes and show that the graph corresponding to the KS vector system must be 4-colourable, as claimed. □

Remark 1. *The proof can straightforwardly be extended to countable graphs representing KS vector systems.*

Since all embeddable graphs are square-free, we are only interested in the 101-colourability of square-free graphs. This problem is still NP-complete. The proof is almost identical to the NP-completeness proof of the 101-colourability problem for general graphs. We show how to reduce instances of 3-COLOR to the problem of checking a square-free graph for 101-colorability. The construction is as before, except that instead of edges joining two

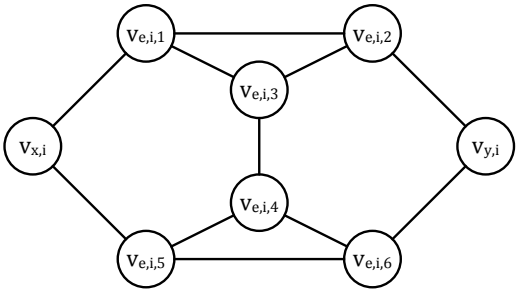

Figure 3.3: The square-free gadget used to connect vertices in G'. The leftmost ($v_{x,i}$) and rightmost ($v_{y,i}$) vertices cannot both be coloured 0.

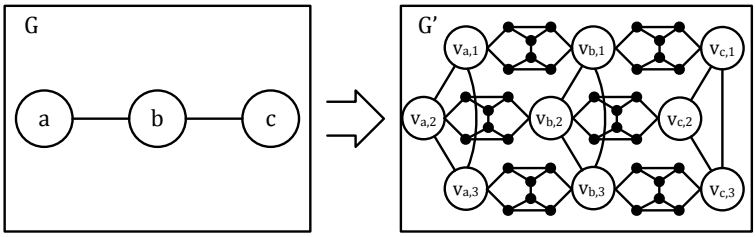

Figure 3.4: The construction of G' from G.

vertices in the newly constructed graph that correspond to different vertices of the original graph, we use a square-free 'gadget' with the appropriate properties.

Theorem 3. *The square-free 101-colourability problem is NP-complete.*

Proof. We know that checking a square-free graph for 101-colourability is a problem in NP, because checking a general graph for 101-colourability is in NP by Thm. 2.

We show how to reduce an instance of 3-COLOR to checking 101-colourability of a square-free graph. Given a graph $G = (V, E)$, construct $G' = (V', E')$. Before we give a formal definition of V' and E', it will be useful to illustrate the shape of the new graph using an example (see Fig. 3.4). Note that the shape of G' is like shape of G' in the construction in the NP-completeness proof of 101-COLOR (Thm. 2), except that the edges connecting v_i and u_i ($v, u \in V$) are now replaced by a special square-free gadget (as shown in Fig. 3.3).

The formal construction looks as follows. We have

$$V' = \left(\bigcup_{x \in V} \{v_{x,1}, v_{x,2}, v_{x,3}\} \right) \cup \left(\bigcup_{e \in E} \left(\bigcup_{i \in \{1,2,3\}} \{v_{e,i,1}, \ldots, v_{e,i,6}\} \right) \right)$$

and

$$E' = \left(\bigcup_{x \in V} \{\{v_{x,1}, v_{x,2}\}, \{v_{x,2}, v_{x,3}\}, \{v_{x,1}, v_{x,3}\}\} \right) \cup \left(\bigcup_{e \in E} \left(\bigcup_{i \in \{1,2,3\}} E_{e,i} \right) \right) \ ,$$

where for $e = \{x, y\}$ each $E_{e,i}$ is a set consisting of the 11 edges:

$$\begin{aligned} E_{e,i} = \big\{ &\{v_{x,i}, v_{e,i,1}\}, \{v_{x,i}, v_{e,i,5}\}, \{v_{y,i}, v_{e,i,2}\}, \{v_{y,i}, v_{e,i,6}\}, \\ &\{v_{e,i,1}, v_{e,i,2}\}, \{v_{e,i,5}, v_{e,i,6}\}, \{v_{e,i,1}, v_{e,i,3}\}, \{v_{e,i,2}, v_{e,i,3}\}, \\ &\{v_{e,i,4}, v_{e,i,5}\}, \{v_{e,i,4}, v_{e,i,6}\}, \{v_{e,i,3}, v_{e,i,4}\} \big\} \ . \end{aligned}$$

Hence $|V'| = 3 \cdot (|V| + 6|E|)$ and $|E'| = 3 \cdot |V| + 11 \cdot |E|$, i.e. the increase in size is still polynomial (in fact $\mathcal{O}(|V| + |E|)$).

We now have to establish the following three facts:

1. G' is square-free.
2. If G is 3-colourable, then G' is 101-colourable.
3. If G' is 101-colourable, then G is 3-colourable.

We split the proofs of those three facts into three lemmas.

Lemma 3. *G' is square-free.*

Remark 2. *This fact is actually obvious by inspection. The formal proof is included for completeness.*

Proof. Note that G' contains a square iff there exist vertices a and b with more than one neighbour in common. Call the vertices $v_{x,i}$ for $x \in V$, $i \in \{1, 2, 3\}$ 'triangle vertices' (because they all appear in triangles). We call the remaining vertices ($v_{e,i,j}$ for $e \in E$, $i \in \{1, 2, 3\}$, and $j \in \{1, \ldots, 6\}$) edge vertices. We show that neither a nor b can be a

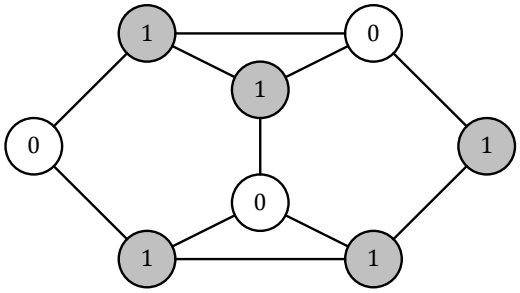

Figure 3.5: Colouring the gadget if the end-vertices have different colours (0 = white, 1 = grey).

triangle vertex or an edge vertex. Since the entire graph consists only of triangle and edge vertices, we deduce that G' does not contain any square.

First, suppose that a is a triangle vertex. Then b cannot lie on the same triple of triangle vertices, because any two triangle vertices on the triple share only one common neighbour. Furthermore, b cannot be a triangle vertex belonging to a different vertex in G, because by construction any two triangle vertices in G' corresponding to different vertices in G are connected only by paths containing at least two edge vertices. This is because every edge in G is replaced by a gadget in G'. Any path across a gadget contains at least two edge vertices. We deduce that at least one of a, b must be an edge vertex.

Suppose now that a is an edge vertex. Since the gadget is square-free, a and b must lie on different gadgets. But a must share two neighbours with b. Both neighbours must also be vertices on the same gadget as a, because a is 'inside' the gadget. By inspection, we find that the only two vertices on the gadget with neighbours outside the gadget are $v_{x,i}$ and $v_{y,i}$ (see Fig. 3.3). However, those two do not have any edge vertex as a common neighbour. But a would have to be a common neighbour of both, so we arrive at a contradiction.

Thus, by symmetry, we have shown that neither a nor b can be edge vertices. Also, we have shown that not both a and b can be triangle vertices. Consequently, there exist no two vertices a and b in G' which share two common neighbours, i.e. G' is indeed square-free. □

Lemma 4. *If G is 3-colourable, then G' is 101-colourable.*

Proof. Suppose that G is 3-colourable. Lemma 1 tells us that G' would be 101-colourable

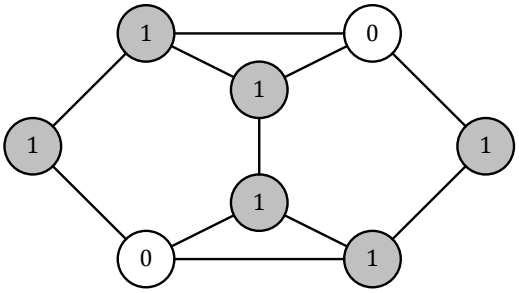

Figure 3.6: Colouring the gadget if the end-vertices are coloured 1 (0 = white, 1 = grey).

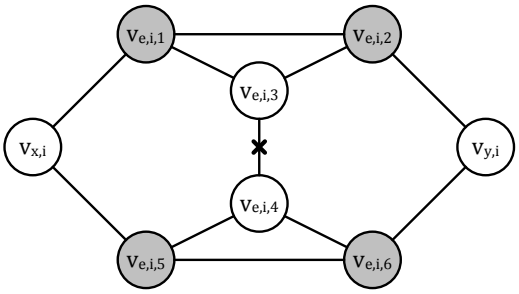

Figure 3.7: Not both end vertices of a gadget can be coloured 0. A colour 0 is represented by a white vertex, a colour of 1 by a grey vertex.

if the gadgets were replaced by regular edges. We use the colouring constructed in Lemma 1 and show that the additional vertices (introduced by the gadgets) can also be coloured. Note that each of those additional vertices is connected to at most one vertex that is already coloured. Thus, we only have to show that there exists a colouring for the gadget in Fig. 3.3 for the following three situations:

1. $v_{x,i}$ is coloured 0 and $v_{y,i}$ is coloured 1.
2. $v_{x,i}$ is coloured 1 and $v_{y,i}$ is coloured 0.
3. $v_{x,i}$ is coloured 1 and $v_{y,i}$ is coloured 1.

The first two situations are the same by symmetry. Figures 3.5 and 3.6 give the colourings for situations 1 and 3, respectively.

□

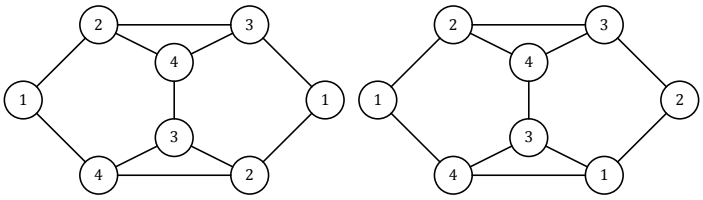

Figure 3.8: 4-colouring a gadget

Lemma 5. *If G' is 101-colourable, then G is 3-colourable.*

Proof. Let $c' : V' \mapsto \{0, 1\}$ be a 101-colouring of G'. For $x \in V$ exactly one of $c'(v_{x,1})$, $c'(v_{x,2})$, $c'(v_{x,3})$ is 0. We define a 3-colouring $c : V \to \{1, 2, 3\}$ by $c(x) = i \Leftrightarrow c'(v_{x,i}) = 0$. It remains to show that c is indeed a valid 3-colouring. We show that if $e = \{x, y\}$ is an edge of G, then $c'(v_{x,i}) \neq 0$ or $c'(v_{y,i}) \neq 0$ for $1 \leq i \leq 3$. Indeed, as Fig. 3.7 shows, it is impossible to colour the gadget between $v_{x,i}$ and $v_{y,i}$ in such a way that 0 is assigned to both vertices: If $v_{x,i}$ and $v_{y,i}$ are both coloured 0, then their four neighbours $(v_{e,i,1}, v_{e,i,2}, v_{e,i,5}, v_{e,i,6})$ must be coloured with colour 1. This forces the remaining vertices $v_{e,i,3}$ and $v_{e,i,4}$ to have colour 0 – a contradiction, since they are adjacent. □

Lemmas 3, 4, and 5 imply that deciding the 101-colourability of a square-free graph is NP-hard. Together with the fact that this problem lies in NP, this means that deciding the 101-colourability of a square-free graph is NP-complete, as claimed. □

Remark 3. *It is not difficult to see that the constructed graph G' is also 4-colourable. Indeed, we can 4-colour each gadget in such a way that both end vertices $v_{x,i}$ and $v_{y,i}$ have the same or different colours, as shown in Fig. 3.8. Using colourings like this (possibly permuting the colours appropriately), we can 4-colour G' by colouring $v_{x,i}$ with colour i for all $v \in V$.*

The reader may be interested to see the following

Proposition 4. *The gadget in Fig. 3.3 is the unique smallest square-free graph with the property that there exist two non-adjacent vertices which cannot both be coloured* 0 *in a valid 101-colouring.*

Proof. (*computer-aided*) By exhaustive enumeration using the methods described later in this thesis. □

3.1 Embedding Polynomials

Suppose that $G = (V, E)$ is a simple undirected graph with vertex set $V = \{v_1, \ldots, v_n\}$. We are interested in finding a direction set $\mathcal{K}$ whose graph representation is isomorphic to G. The intention, of course, is to find a graph that is not 101-colourable and to determine a corresponding direction set $\mathcal{K}$. Such a set would be a KS vector system, since no 101 function could be defined on $\mathcal{K}$. Therefore, we are actually only interested in finding a direction set $\mathcal{K}$ with the property that G is a subgraph of the graph representation of $\mathcal{K}$.

Let $\mathbf{x_i}$ be the vector in $\mathcal{K}$ corresponding to vertex v_i in G. There are now three conditions that the $\mathbf{x_i}$ have to fulfil:

1. $\mathbf{x}_i \neq \mathbf{0}$ for all $i \in \{1, \ldots, n\}$.
2. $(v_i, v_j) \in E \Rightarrow \mathbf{x}_i \cdot \mathbf{x}_j = 0$ for all $i, j \in \{1, \ldots, n\}$.
3. $\mathbf{x}_i \times \mathbf{x}_j \neq \mathbf{0}$ for all $i, j \in \{1, \ldots, n\}, i \neq j$.

As soon as $\mathcal{K}$ contains more than one element, the first and the third conditions can be summarized by saying that no two vectors may be collinear. In the sequel, we will refer to the conjunction of conditions 1 and 3 as the 'collinearity constraint'. Condition 2 will be referred to as the 'orthogonality constraint'.

The orthogonality constraint can be formulated nicely as a system of polynomial equations. A direction set $\mathcal{K}$ whose graph representation is isomorphic to G must satisfy all of the given equations. The system contains one equation for each edge of G. The equation for a given edge $e = \{v_i, v_j\} \in E$ is

$$\mathbf{x_i} \cdot \mathbf{x_j} = \mathbf{0} \ , \tag{3.1}$$

which can be rewritten as

$$x_i(1)x_j(1) + x_i(2)x_j(2) + x_i(3)x_j(3) = 0 \ . \tag{3.2}$$

Here we write $x_i(k)$ for the k^{th} component of vector $\mathbf{x}_i = (x_i(1), x_i(2), x_i(3))^{\mathrm{T}}$.

If this system of equations has a single real solution, then it has infinitely many solutions: First of all, a solution can be scaled by any non-zero scalar to arrive at another solution. In fact, each of the vectors in the direction set can be scaled by a separate non-zero scalar. This can easily be prevented by adding an additional equation for each vertex v_i in the graph:

$$x_i(1)^2 + x_i(2)^2 + x_i(3)^2 - 1 = 0 \tag{3.3}$$

This forces all the vectors in the direction set to be of unit length.

Secondly, any solution can be rotated around the origin to arrive at more solutions. This can be prevented by fixing the directions corresponding to the two vertices of one edge (assuming that the graph contains an edge). If the edge is part of a triangle, then the direction corresponding to the third vertex of the triangle is also determined.

Fortunately, it is known that the problem is decidable. In fact, the conditions can be expressed as a formula in the existential first-order theory of the reals. Deciding the truth of such a formula is known to be a problem in PSPACE ([2], [22]). However, we require an efficient algorithm, which can decide the embeddability of a graph within seconds.

A first, naïve approach to finding an embedding was to ask the software package Wolfram Mathematica 7.0 for a symbolic solution to the system of equations. Unfortunately, the command never terminated, even for relatively small systems.

Thereafter, a significant amount of time was spent on learning about and trying out methods for finding embeddings of graphs. Personally, I believe that finding a reliable and efficient method for determining whether a graph is embeddable is the key to finding smaller KS vector systems, or showing that none exist. Unfortunately, none of the methods under consideration were refined to a point where they could usefully be applied to all the graphs that were encountered.

The next sections cover the methods that were applied, their benefits and their drawbacks. Finally, we present some surprisingly small graphs whose embeddability was hard or impossible to decide using the described methods.

3.2 Embedding via Numerical Minimization

Wolfram Mathematica offers a function, called `NMinimize`, which uses various numerical methods in order to find the global minimum of a given function in multiple variables.

In order to use `NMinimize`, the system of equations encoding the orthogonality conditions must be converted into a single function which reaches its known minimum iff the system has a solution. This is done by summing the squares of all the left-hand sides of the equations. All terms in the sum are non-negative. Hence, the sum has minimum 0 iff the system of equations has a solution. Note that this does not take into account the collinearity condition.

Let $G = (V, E)$ be a graph with $|V| = n$ vertices. Formally, the function being minimized is given by:

$$f(\mathbf{x}_1, ..., \mathbf{x}_n) = \sum_{i=1}^{n} \left(1 - |\mathbf{x}_i|^2\right)^2 + \sum_{\{i,j\} \in E} (\mathbf{x}_i \cdot \mathbf{x}_j)^2 \tag{3.4}$$

Note that we do not fix any coordinates in advance. This leaves a higher-dimensional set of minima in the function (if the graph has an embedding). Whether or not this is advantageous was not investigated.

Section 7.1 in the Appendix contains the Mathematica input which finds an embedding for Conway's KS system.

3.2.1 Advantages

This provides a very quick method of finding the numerical approximation to an embedding of a graph. The function call terminates within about one second, which makes this method relatively fast compared to the other methods – especially when looking at the worst cases. The method can deal with all graphs in the range that we are interested in. Converting a graph to the corresponding input for Mathematica is also quite straight-forward, i.e. the implementation effort is low.

3.2.2 Drawbacks

The algorithm underlying `NMinimize` is not guaranteed to find the global minimum of the function. It is therefore impossible to use this method in order to decide that a graph has no embedding. Furthermore, even if an embedding is found, the coordinates are not found accurately, although the solution's accuracy may be increased. Sometimes it is possible to find the exact coordinates by rotating the discovered embedding until three orthogonal vectors are aligned with the unit vectors. The coordinates of the remaining vertices may then be close to some "familiar" values, like $\frac{1}{\sqrt{2}}$ or $\frac{1}{\sqrt{3}}$. Using those values for the coordinates, one can again verify that they solve the system of equations.

Sometimes the method finds only embeddings that have collinear vectors, even when run with many different seeds for the random number generator used by the implementation of `NMinimize`. One would like to conclude that, with high probability, no embedding exists that does not contain pairs of collinear vectors. However, it is probably very complicated (even knowing all the implementation details) to derive bounds for the probability of certain minima appearing.

3.3 Embedding with the 'Box Grid'

Conway's KS system lies on a very regular grid, as shown in Fig. 2.4. The grid (including the points that do not belong to Conway's system) corresponds to the direction set containing the vectors with integer coordinates on the surface of the cube $[-2, 2]^3$. For every such vector $\mathbf{v}$, there is also an exactly opposite vector $-\mathbf{v}$ with integer coordinates that lies on the surface of the cube. Obviously, we consider only one of those two collinear vectors. Fig. 3.9 shows the box grid on which Conway's system lies. The vectors that belong to the direction set are marked by small circles. There are 49 such vectors on the entire grid, while Conway's system makes use of only 31 of them.

We can generalize the grid and introduce a *grid parameter* N: The direction set corresponding to a grid with positive integer grid parameter N corresponds to the (non collinear) vectors with integer coordinates that lie on the surface of the cube $[-N, N]^3$. Hence, Conway's KS system is embeddable on the box grid with grid parameter $N = 2$. Obviously, it

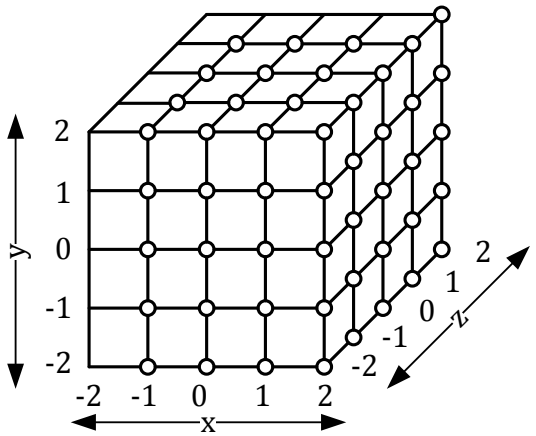

Figure 3.9: The box grid underlying Conway's KS system.

is also embeddable on the box grid with grid parameter $N = 4$, which contains the same directions as the box grid with parameter $N = 2$, and a few more.

In order to check whether a given graph is embeddable, one can look for an embedding on a box grid. In fact, a graph G is embeddable if it is the subgraph of the graph representation of the direction set corresponding to a box grid.

3.3.1 Advantages

Finding an embedding on a box grid gives an exact symbolic representation of the KS vector system. At the same time, it guarantees that the embedding is valid. Checking embeddability on the box grid with grid parameter $N = 2$ can be done very quickly, so the method provides a quick way of embedding graphs that are easy to embed.

3.3.2 Drawbacks

Deciding subgraph isomorphism is unfortunately NP-complete in general [8]. It seems that this should remain true if only square-free graphs are considered. This was not investigated in detail, because it is not central to this work. It is less obvious if checking whether a given graph is embeddable in a box grid with given grid parameter N is NP-complete. However, no efficient algorithm was found. The check is infeasible, even for small values of

N ($N = 32$, say).

Furthermore, it is not clear whether every embeddable finite graph has an embedding on some box grid. If this was the case, then this would put the problem of finding an embedding of a given graph into a discrete setting. The following straight-forward propositions work towards establishing this result. Unfortunately, the result itself was not obtained.

Proposition 5. *A finite graph G has an embedding on some box grid iff it has an embedding on the surface of the unit cube $[-1,1]^3$ with only rational coordinates.*

Proof. First, we show that we can construct an embedding with rational coordinates on the surface of the unit cube $[-1,1]^3$ from an embedding on the box grid with grid parameter N. Let $\mathcal{K} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be an embedding of G on the box grid with parameter N. All the coordinates of the $\mathbf{x}_i$ are integers. Consider the direction set $\mathcal{K}' = \{\frac{1}{N}\mathbf{x}_1, \ldots, \frac{1}{N}\mathbf{x}_n\}$. All vectors in $\mathcal{K}'$ have rational coordinates that lie on the surface of $[-1,1]^3$.

On the other hand, suppose that $\mathcal{K}' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_n\}$ is an embedding on the surface of $[-1,1]^3$ with rational coordinates, say $\mathbf{x}_i = \left(\frac{p_i(1)}{q_i(1)}, \frac{p_i(2)}{q_i(2)}, \frac{p_i(3)}{q_i(3)}\right)^{\mathrm{T}}$. Let k be the least common multiple of all the $q_i(j)$. Such a k exists, because G is finite by assumption. Then $\mathcal{K} = \{k\mathbf{x}'_1, \ldots, k\mathbf{x}'_n\}$ contains only integer coordinates on the surface of the box grid with parameter $N = k$. □

From now on, we refer to a vector in a direction set as *rational* iff all of its coordinates are rational numbers. A vector that is not rational is called *irrational*. Hence, $\mathbb{Q}^3$ denotes the set of rational vectors.

Proposition 6. *Let $\mathbf{x}$ and $\mathbf{y}$ be rational vectors (not collinear) on the surface of the unit cube $[-1,1]^3$. Then the two vectors orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ that lie on the surface of $[-1,1]^3$ are also rational.*

Proof. The two orthogonal vectors are $\pm k(\mathbf{x} \times \mathbf{y})$, where $k = \|\mathbf{x} \times \mathbf{y}\|_\infty^{-1}$, i.e. we divide through by the coordinate with greatest magnitude of $\mathbf{x} \times \mathbf{y}$ to take the orthogonal vector back to the surface of the cube. Note that $\|\mathbf{x} \times \mathbf{y}\|_\infty^{-1} > 0$, because $\mathbf{x}$ and $\mathbf{y}$ are not collinear by hypothesis.

Let $\mathbf{z} = (z(1), z(2), z(3))^{\mathrm{T}} = \mathbf{x} \times \mathbf{y}$. The coordinates of $\mathbf{z}$ are:

$$\begin{aligned} z(1) &= x(2)y(3) - y(2)x(3) \\ z(2) &= x(3)y(1) - y(3)x(1) \\ z(3) &= x(1)y(2) - y(1)x(2) \ , \end{aligned}$$

i.e. all coordinates are rational. But then k is rational, too. It follows that the two vectors $\pm k \cdot \mathbf{z}$ are also rational, as claimed. □

Corollary 1. *In an embedding on the surface of* $[-1, 1]^3$*, either none or at least two of the vectors assigned to the vertices of a triangle are irrational.*

Corollary 2. *In an embedding on the surface of* $[-1, 1]^3$*, at most one of the vectors orthogonal to an irrational vector is rational.*

Note that we can choose any two adjacent vertices of a given graph and assume the corresponding vectors are rational. Unfortunately, this is far from sufficient to show that all vectors can be rational. However, since no emabeddable graph was found that has no embedding on some box grid, we make the following conjecture:

Conjecture 1. *Every finite, embeddable graph has an embedding on some box grid with finite grid parameter* N*.*

One might wonder whether it is always possible to use vectors with rational coordinates that lie on the unit sphere for an embedding. Fortunately, the answer to this question is known: a 101-function can be defined on the rational points of the unit sphere. Thus, any graph that is not 101-colourable cannot be embeddable using only rational coordinates on the unit sphere.

The following proof is due to D. A. Meyer [15] and is based a result by Godsil and Saks [9] and a theorem of Hales and Straus [11].

Theorem 4. *The rational vectors on the unit sphere are 101-colourable.*

Proof. We give an explicit colouring of the rational vectors on the unit sphere. Let $\mathbf{x} = \left(\frac{p_1}{q_1}, \frac{p_2}{q_2}, \frac{p_3}{q_3}\right)^{\mathrm{T}}$ be a rational vector on the unit sphere ($\gcd(p_i, q_i) = 1$), and k the least

common multiple of q_1, q_2, and q_3. Let $\hat{\mathbf{x}} = k \cdot \mathbf{x}$ be one of the two shortest collinear vectors with integer coordinates (which of the two vectors will not matter for the construction). We denote the components of $\hat{\mathbf{x}}$ by $\hat{x}(1)$, $\hat{x}(2)$, and $\hat{x}(3)$. Since the original vector $\mathbf{x}$ satisfies:

$$\left(\frac{p_1}{q_1}\right)^2 + \left(\frac{p_2}{q_2}\right)^2 + \left(\frac{p_3}{q_3}\right)^2 = 1 \ ,$$

we also know that the components of $\hat{\mathbf{x}}$ satisfy

$$\hat{x}(1)^2 + \hat{x}(2)^2 + \hat{x}(3)^2 = k^2 \ .$$

The square of an odd number is congruent to 1 mod 4, since $(2n+1)^2 = 4n^2 + 4n + 1$. Similarly, the square of an even number is congruent to 0 mod 4. Consequently, if k is even, then all of $x(1)$, $x(2)$, and $x(3)$ must be even, or else their squares can only sum to 1, 2, or 3 mod 4. If all of $x(1)$, $x(2)$, and $x(3)$ are even, then $\hat{\mathbf{x}}$ cannot be the shortest collinear vector with integer coordinates. Thus, we conclude that k must be odd. At the same time, exactly one of $\hat{x}(1)$, $\hat{x}(2)$, and $\hat{x}(3)$ must be odd.

Consider now the function

$$f(\mathbf{x}) = \begin{cases} 0 & \text{if } \hat{x}(1) \text{ is odd} \\ 1 & \text{otherwise} \end{cases} \ .$$

We claim that f is a 101-function on the rational vectors on the unit sphere. Indeed, if $\mathbf{x}$ and $\mathbf{y}$ are two orthogonal vectors, then

$$\hat{x}(1)\hat{y}(1) + \hat{x}(2)\hat{y}(2) + \hat{x}(3)\hat{y}(3) = 0 \ .$$

Considering this equation modulo 2, we see that this is only possible if the odd coordinate of $\hat{\mathbf{x}}$ is in a different place than the odd coordinate of $\hat{\mathbf{y}}$. Hence, we deduce that only one of three mutually orthogonal vectors has an odd first integer coordinate in its smallest integer representation, i.e. f is indeed a 101-function. □

3.4 Embedding Using Numerical Continuation

Looking for more reliable methods to solve the systems of equations, Joël Ouaknine and I were pointed to a software called Bertini [1]. I am indebted to Prof. C. W. Wampler, one of the authors of Bertini, who explained very patiently how to use the software and who produced the input file which was used to show (numerically) that a subgraph of the uncolourable 17 vertex graph is not embeddable (see Chapter 4).

We give a brief outline of the algorithm underlying Bertini. A very detailed treatment can be found in the book by A. J. Sommese and C. W. Wampler [24].

The general idea of homotopy continuation is the following: Given a system of polynomial equations $f_i(\mathbf{x}) = 0$ in the variables $\mathbf{x}$ that we are trying to solve, find another system of equations $g_i(\mathbf{x}) = 0$ with known solutions and of matching degrees. Consider the system of equations $h_i(\mathbf{x}, t) = t \cdot g_i(\mathbf{x}) + (1 - t)f_i(\mathbf{x}) = 0$. We know the solution of $h_i(\mathbf{x}, t) = 0$ at $t = 1$ and we seek a solution at $t = 0$. Assuming that "nothing goes wrong" (which is a strong assumption, see the book for details), we can numerically track the paths of the roots of $h_i(\mathbf{x}, t)$ as t goes from 1 to 0. This path tracking can be done, for example, by using Euler prediction and Newton correction steps. In order for all of this to work, we have to work in the complexes instead of just the reals. If the roots of the initial system of equations are isolated (i.e. the solution set has dimension 0), then this method (when combined with several enhancements, e.g. the *Gamma Trick*, which makes use of a random number generator in order to avoid bad start systems with probability 1) will produce this set of isolated roots with mathematical probability 1. Of course, the actual probability may be slightly lower than 1 because of the finite precision in the floating point numbers on a computer.

The system of orthogonality equations has a very nice property. Each equation has the form

$$\mathbf{x}_i \cdot \mathbf{x}_j = 0 \ .$$

Hence, all equations are homogeneous in each of the $\mathbf{x}_i$: if we have a solution that satisfies all equations, we can scale each $\mathbf{x}_i$ by an arbitrary non-zero λ_i and we still have a solution. Thus, we can identify all collinear, non-zero vectors and work in the projective space $\mathbb{P}^2$.

Of course, this was clear from the start. We are looking for a set of directions, rather than actual points in $\mathbb{R}^3$. Working in $\mathbb{P}^2$ also means that we do not need the additional equations forcing the solution vectors to have unit length.

Bertini allows the definition of homogeneous variable groups. Those are groups of variables with respect to which the system of equations is homogeneous. In the case of our problem, the sets $\{x_i(1), x_i(2), x_i(3)\}$ form homogeneous variable groups. The use of such variable groups helps reducing the root count of the start system: fewer of the paths will lead to singular roots.

The system of orthogonality equations can only have isolated solutions if we fix the coordinates of the vectors corresponding to two adjacent vertices (or, if the graph has a triangle, we can fix the coordinates of the vectors corresponding to the triangle's vertices). This eliminates some variables and simplifies a few equations. Furthermore, if we fix a particular vector to have coordinates $(1, 0, 0)^{\mathrm{T}}$, say, then all orthogonal vectors must have the first coordinate set to 0. This eliminates more variables and simplifies further equations. An example input file for Bertini is shown in Sect. 7.2 of the appendix.

3.4.1 Advantages

Especially when a system of equations has isolated solutions, Bertini is practically guaranteed to find those solutions. The accuracy of the numerical computations may have to be increased, or one might end up in a "probability 0" case (running Bertini again will fix this), but mathematically, the method works almost surely.

Bertini is relatively fast on small systems of equations. For larger systems, a method called "regeneration" can be used to speed up the root finding process. However, I was unable to solve the system of equations resulting from Conway's KS system.

3.4.2 Drawbacks

Although Bertini works well if the graph being embedded has exactly the right number of edges and only isolated solutions, it is more difficult to deal with the other cases. A graph $G = (V, E)$ has exactly the right number of edges if $E = 2 \cdot |V| - 3$ (each vector in the embedding has two degrees of freedom and by fixing two adjacent vertices, we remove three

degrees of freedom). Note that the right number of equations does not necessarily mean that the solution set is 0-dimensional.

If a system has fewer equations or positive dimensional solution sets, then one has to intersect the higher-dimensional solution sets with the reals in order to check whether the result is non-empty. This is currently not possible with Bertini. An algorithm for finding the real points on a complex curve is known [14] and may be in a future release of Bertini.

If the system of orthogonality equations has more equations than unknowns, then it may still be possible to find an embedding using Bertini. For example, if there are m equations in n unknowns ($m > n$), then one can consider n equations chosen uniformly at random. With luck, those equations have isolated solutions. One still has to verify that the solutions also satisfy the remaining equalities. Another possibility is taking n random linear combinations of the n equations (where the coefficient of each equation is chosen uniformly between 0 and 1).

Unfortunately, the process of solving a large system of equations can be rather slow. For example, using a random selection of the orthogonality equations that belong to Conway's KS system, Bertini ran for over a week and was tracking 122866 paths when the execution was aborted (the run was using the "regeneration" feature, where new equations are brought in one at a time).

If Bertini actually finds an embedding (as a real root of the system of equations), then this does not constitute a computational proof of the fact that the system has an embedding. Due to the finite precision in the solution and the computations of Bertini (even if adaptive precision is used), the embedding will only satisfy the equations within a (very small and configurable) error margin. Of course, such a solution is a very strong indication of the fact that such an embedding exists and it may be possible to "guess" the correct values of some of the variables and then use a symbolic solver to determine the remaining values symbolically.

3.5 Embedding Using Interval Arithmetic

Interval Arithmetic is the method used by Pavičić et al. in [17]. I thank Dr Jean-Pierre Merlet, one of the authors of the article and author of the interval arithmetic library ALIAS [21], for sharing some of his experience in embedding KS vector systems, and for helping me use the library. The results found in this thesis were obtained using a custom interval analysis solver based on the interval analysis library PROFIL / BIAS [20].

The idea underlying interval arithmetic is that the real values of the variables are replaced with intervals. For example, suppose that we know that $x \in [-1, 1]$ and $y \in [-1, 0]$. Then we can already decide that $x + y \in [-2, 1]$ and that the equation $x + y = 2$ has no solution on the given ranges. Here, we have used the *interval evaluation* of the + operator: if $x \in [\underline{x}, \overline{x}]$ and $y \in [\underline{y}, \overline{y}]$, then $x + y \in [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$. Similarly, an interval evaluation for multiplication can be defined (and, of course, for many other operators). In fact, when implemented on a computer, such interval operators can even take into account the rounding errors that are introduced by the finite precision arithmetic.

A collection of intervals, one interval for each variable involved in a system of equations, is called a *box*. Given such a box $\mathcal{B}$, we may already be able to decide that within this box, there can be no embedding of the graph that we are trying to embed. For example, it may be impossible for one of the scalar products to evaluate to 0.

For the purpose of finding an embedding $\mathcal{K} = \{\mathbf{x}_i\}_{i=1\ldots n}$, the initial intervals can be set as follows:

$$\begin{aligned} x_i(1) &\in [-1, 1] \\ x_i(2) &\in [-1, 1] \\ x_i(3) &\in [0, 1] \end{aligned}$$

Of course, we can already fix some of the vertices or coordinates at the beginning, as usual. The initial intervals are not likely to indicate that no embedding exists. Furthermore, if an embedding exists, we would like to get a close approximation to the solution. There

are several methods that can decrease the width of some of the intervals, two of which we outline here (the 2B and the 3B method). If the width of the intervals cannot be decreased any further, it is necessary to split the current box into two boxes and treat each of the boxes separately. This process is called *bisection*.

3.5.1 The 2B Method

Suppose that one of the orthogonality equations is

$$x_1(1)x_2(1) + x_1(2)x_2(2) + x_1(3)x_2(3) = 0 \ .$$

Let $\tilde{x}_i(k)$ denote the current interval for variable $x_i(k)$. If $\tilde{x}_2(1)$ does not contain 0, then it must be the case that

$$x_1(1) \in -\frac{\tilde{x}_1(2)\tilde{x}_2(2) + \tilde{x}_1(3)\tilde{x}_2(3)}{\tilde{x}_2(1)} \ ,$$

where the operators on the right hand side are interval arithmetic operators. This may allow us to decrease the width of interval $\tilde{x}_1(1)$. We may even be able to deduce that there is no solution of the equation inside the current box. This method is called the "2B Method" and it can similarly be applied to all the other variables involved in the equation.

3.5.2 The 3B Method

Given a box $\mathcal{B}$, we may be able to determine that it cannot contain a solution. This will most likely be the case if some of the intervals have a particularly small width. Suppose that $\tilde{x}_1(1) = [\underline{x}_1(1), \overline{x}_1(1)]$ is the interval for variable $x_1(1)$ in box $\mathcal{B}$. Suppose further that we can determine that $\mathcal{B}$ contains no solution if we change $\tilde{x}_1(1)$ to be $[\underline{x}_1(1), \underline{x}_1(1) + \varepsilon]$. Then we can replace $\tilde{x}_1(1)$ by the interval $[\underline{x}_1(1) + \varepsilon, \overline{x}_1(1)]$. We can use binary search to determine a good value for ε. This method is called the "3B Method". It can also be applied to the upper bound of an interval, and to all intervals involved in the problem.

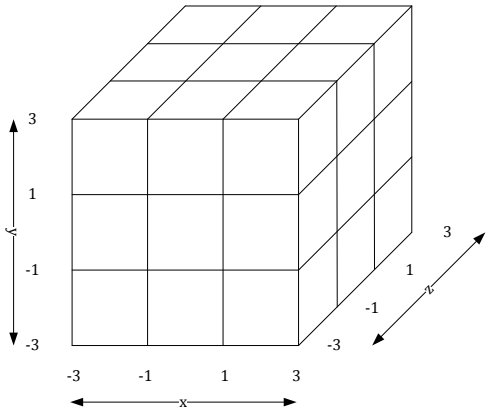

Figure 3.10: Initial intervals for the interval solver.

3.5.3 Equations for the Interval Solver

There are many different ways of formulating the problem of embedding a particular graph for an interval arithmetic solver. One possibility is to take the orthogonality equations and one equation per vector in the embedding, saying that the length of the vector should be 1.

Another interesting approach uses as start intervals the squares that are visible on the cube shown in figure 3.10. A recursive backtracking algorithm will find 'valid' assignments of vectors to the intervals. Here, valid means that the orthogonality equations are still satisfiable. During this process, the 2B and 3B methods are already applied to decrease the sizes of the intervals. However, no bisection takes place. Once a valid assignment of the starting intervals has been found, the full interval arithmetic solver algorithm is applied to find actual solutions within the intervals. This method benefits from the fact that every vector has only two components that are intervals: the third component is fixed, because the vector must lie on one side of the cube. This reduces the number of intervals by 1/3. However, we also remove all the equations saying that the length of a vector must be 1. This leaves fewer equations to which the 2B method can be applied.

Yet another way of phrasing the orthogonality conditions is via distance equations: If we think of the vectors in an embedding as points on the unit sphere, then the distance of any point to the origin must be 1, while the distance between two points corresponding to two adjacent vertices in the graph must be $\sqrt{2}$. In an interval arithmetic solver, those

equations can be used in addition to the regular orthogonality equations in order to provide more equations for the 2B method. Whether or not this is beneficial was not investigated.

3.5.4 Advantages

In certain situations, interval analysis can provide a complete, computational proof (not just a numerical proof) of the fact that an embedding does not exist: if there are no real solutions of the orthogonality equations, then there is a set of (possibly many, small) boxes which cover the entire space of potential solutions, and in each of which the system of equations provably has no solution.

The algorithms underlying a simple interval analysis solver are rather easy to understand and can be implemented within a reasonable amount of time. This can probably not be said about the algorithms underlying numerical continuation.

Finally, interval analysis is often quite efficient in finding solutions for an embeddable graph, although there are some cases where this is not the case.

3.5.5 Drawbacks

It is difficult to deal with embeddings that have solutions where two vectors are collinear: the algorithm has to determine whether or not two vectors within a given solution are collinear or not (and possibly discard the solution). If the vectors are required to be of unit length, one can check whether or not the magnitude of the dot product between two vectors (i.e. the sine of the angle between the two vectors) is greater than a certain threshold. For example, Pavičić et al. [17] use 0.99 as the threshold. This corresponds to a minimum allowed angle of roughly 8 degrees between any two vectors. This means that any solutions with vectors that are almost collinear will be (incorrectly) discarded. It seems that whenever there is a valid embedding, then there is an embedding in which the directions are reasonably far apart (considering the number of vectors in the system). However, there is no immediate reason to assume that no graphs exist which only admit embeddings that contain at least one pair of vectors that make an angle smaller than the given threshold. In fact, all of the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_k$ corresponding the the k neighbours $u_1, \ldots, u_k$ of a vertex v must lie in one plane, namely the plane orthogonal to the vector corresponding to v. If all pairs $\mathbf{y}_i$ and

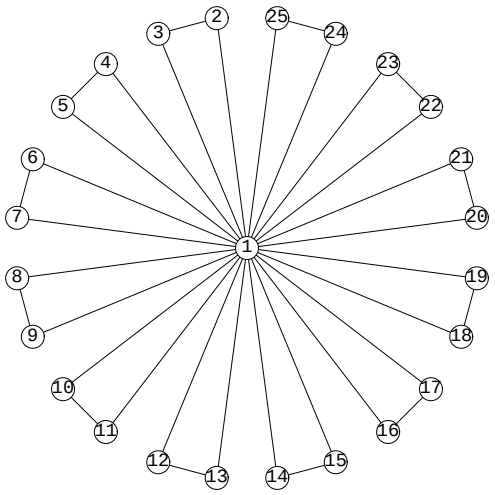

Figure 3.11: A graph that forces the minimum angle between two vectors in the embedding to be less than 8 degrees.

$\mathbf{y}_j$ of vectors ($i \neq j$) must have a dot product with magnitude less than 0.99, then there can be at most 22 vectors in this plane ($\pi / \cos^{-1}(0.99) \approx 22.1959$). This means that the interval arithmetic embedder in [17] may not have found an embedding for any graph that contains the graph shown in Fig. 3.11 as a subgraph.

Clearly, this graph has an embedding: if $\mathbf{x_i}$ is the vector corresponding to the vertex labelled i in Fig. 3.11, then we can set

$$\begin{aligned} \mathbf{x}_1 &= (1,0,0)^{\mathrm{T}} \\ \mathbf{x}_{2i} &= \left(0, \sin\left(i \cdot \frac{\pi}{24}\right), \cos\left(i \cdot \frac{\pi}{24}\right)\right)^{\mathrm{T}} \\ \mathbf{x}_{2i+1} &= \left(0, \cos\left(i \cdot \frac{\pi}{24}\right), -\sin\left(i \cdot \frac{\pi}{24}\right)\right)^{\mathrm{T}} \quad , \end{aligned}$$

for $1 \leq i \leq 12$. It is not difficult to see that the orthogonality conditions are indeed satisfied, and that the vectors $\mathbf{x}_{2i}$ lie on an arc of 90 degrees. The vectors $\mathbf{x}_{2i+1}$ also lie on an arc of 90 degrees, s.t. they are orthogonal both to $\mathbf{x}_1$ and their orthogonal counter-part on the first arc. (The reader may find it much easier to directly construct an embedding than following this written explanation).

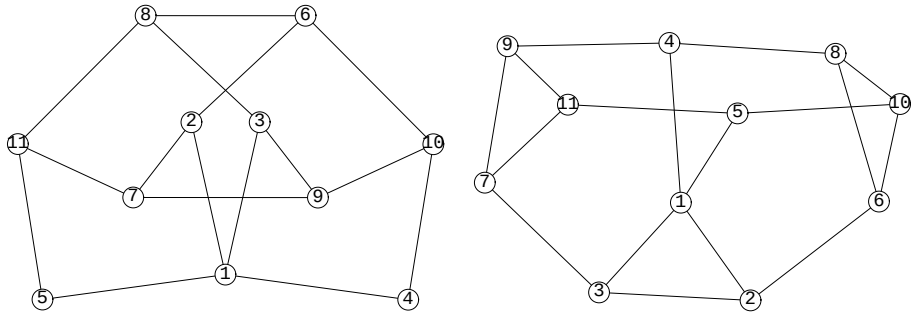

Figure 3.12: Two graphs that were determined to be unembeddable by the interval arithmetic solver.

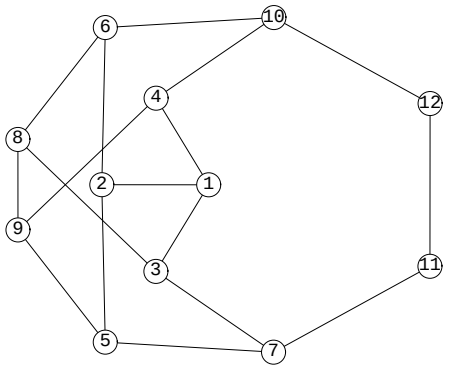

Figure 3.13: A graph with 12 vertices whose embeddability was found difficult to determine.

3.6 Difficult Examples

In this section, we give a few examples of graphs whose embeddability was infeasible or difficult to decide using the methods described above. The embeddability of these graphs can probably be decided without too much effort, when the methods are implemented more carefully and given more time. Nonetheless, they might serve as a starting point for benchmarking automatic embedders.

The first two examples are graphs with 11 vertices (Fig. 3.12). All of their subgraphs are embeddable, but the interval analysis algorithm determined that those graphs have no embeddings.

The third example is a graph with 12 vertices (Fig. 3.13). The interval arithmetic solver was unable to determine the embeddability of this graph within a reasonable amount of

time. It should be remarked that this graph has no triangles and two adjacent vertices which have degree 2. Those features probably make it difficult for the interval analysis algorithm to reduce the widths of the intervals.

3.7 Real Solutions of the Orthogonality Equations

There can be many unwanted real solutions of the orthogonality equations (real solutions in which some of the vectors are collinear), as the following result indicates:

Proposition 7. *Let $G = (V, E)$ be a simple graph with at least one edge, and let N be the number of 3-colourings of G. Let x and y be two adjacent vertices of G. Suppose that we fix the vector corresponding to x in an embedding to be $(1, 0, 0)^T$ and the vector corresponding to y to be $(0, 1, 0)^T$. Suppose further that we require all vectors in the embedding to have a non-negative third coordinate. Then the orthogonality equations of G, together with the condition that the solution vectors have length 1, have at least $\frac{N}{6}$ real solutions.*

Proof. We show how to obtain a distinct embedding of G from every 3-colouring, assuming that the colours of x and y are fixed. If G is coloured using the colours 1, 2, and 3, then every permutation of the three colours yields a new colouring. We will therefore assume that x has colour 1 and y has colour 2. This means we are left with only $\frac{N}{3!} = \frac{N}{6}$ colourings.

Given such a colouring, we assign the vector $(1, 0, 0)^{\mathrm{T}}$ to all the vertices that are coloured 1. All vertices that are coloured 2 and 3 are assigned the vectors $(0, 1, 0)^{\mathrm{T}}$ and $(0, 0, 1)^{\mathrm{T}}$, respectively. Since we have a valid 3-colouring, all edges connect vertices with different colours. The corresponding vectors are therefore orthogonal. This means that we have constructed a valid embedding. Furthermore, distinct colourings (with the colour of x and y being 1 and 2, respectively) result in distinct embeddings.

This shows that there must be at least $\frac{N}{6}$ real solutions to the orthogonality equations.

□

Remark 4. *Not all of those solutions necessarily contain pairs of collinear vectors (but all solutions with more than three directions do). Also, some of the solutions may consist of the same multi-set of vectors.*

A large number of real solutions can make the interval analysis solver very slow: as long as the current box contains multiple solutions, it will be difficult to decrease the width of the box. The algorithm must get lucky and do the right bisection of the box. If all of the real solutions have collinear vectors, then a very large number of boxes must be considered.

An advanced solving algorithm using interval arithmetic could first bisect only on variables that belong to one smallest not 3-colourable subgraph of the given graph. This was not tried out, but it could avoid unnecessary bisections. Note that every graph that is not 101-colourable is also not 3-colourable by Prop. 1.

Chapter 4

Lower Bound

We have seen that every KS vector system can be converted into a simple graph, which is not 101-colourable. Furthermore, we know that such a graph must be C_4-free (square-free). It is therefore natural to ask what the smallest square-free graph is that is not 101-colourable. The size of this graph is a lower bound for the size of the smallest 3-dimensional KS vector system.

The complete answer to this question consists of two parts. On the one hand, we present an uncolourable square-free graph on 17 vertices. On the other hand, we show that no uncolourable square-free graphs on fewer than 17 vertices exist by exhaustively enumerating all such graphs. In this section, we describe the methods employed to establish both results.

4.1 The Smallest Uncolourable Square-Free Graph

4.1.1 Enumerating Square-Free Graphs

At the time of this writing, the On-Line Encyclopaedia of Integer Sequences [23] contains the numbers of square-free connected graphs with up to and including 17 vertices (see Fig. 4.1). The fact that the sequence only contains the number of connected square-free graphs up to 17 vertices indicates that no explicit formula for the sequence is known and that it is in fact somewhat difficult to enumerate all connected square-free graphs on a larger number

n	graphs	n	graphs	n	graphs
1	1	7	57	13	932260
2	1	8	186	14	8596844
3	2	9	740	15	93762704
4	3	10	3389	16	1201732437
5	8	11	18502	17	17992683043
6	19	12	120221		

Figure 4.1: Number of connected square-free graphs on n vertices.

of vertices. The algorithm suggested in this section would probably be powerful enough to obtain the 18^{th} value in this sequence, given one or two weeks of CPU time on a single machine. We attempt to check all connected square-free graphs with at most 17 vertices for 101-colourability. This involves checking almost 18 billion instances of an NP-complete problem (Thm. 3).

In order to do this, one has to enumerate the connected square-free graphs efficiently. We do this by generating the graphs on $n+1$ vertices from the graphs on n vertices by adding in an extra vertex and connecting it to the other n vertices in every possible and valid way. This method works due to the following

Lemma 6. *Every non-empty simple connected graph G contains a vertex that is not a cut vertex.*

Proof. Since G is connected, it has a spanning tree T. A vertex x of T with degree at most 1 (which exists since T is a non-empty tree) can be removed without disconnecting T. Therefore, $T - \{x\}$ is a spanning tree of $G - \{x\}$ proving that x is not a cut vertex of G. □

As a consequence of Lemma 6, we can construct any connected square-free graph on $n+1$ vertices from some connected square-free graph on n vertices by adding an additional vertex. If there was a connected square-free graph in which every vertex is a cut-vertex, then it would not be possible to generate this graph by adding a new vertex to a single connected square-free graph.

Given this result, an obvious way to proceed is to start with a list of graphs on n vertices and generate all graphs on $n+1$ vertices. We add an extra vertex and connect it to the

n existing vertices in all $2^n - 1$ possible ways (there must be at least one incident edge). This method will generate several graphs that are isomorphic. For example, there are two possible ways of generating a path on three vertices from a single edge, depending on which of the two vertices the third vertex is joined to. One solution to the problem is to store all generated graphs in a hash table using a hash function that is invariant over isomorphic graphs. Whenever a collision occurs, an actual isomorphism check must be done. Given the small size of the graphs in question, this approach is actually feasible. Indeed, the check was first done using this method.

Choosing a good hash function is crucial for the performance of the algorithm. However, the computation of the hash value for a given graph should not take too long. If $G = (V, E)$ is a simple undirected graph, then the hash value of G is computed by the hash function h:

$$
\begin{aligned}
e(v) &= \text{the number of neighbours of vertex } v && (4.1)\\
P_n &= \text{the } n^{\text{th}} \text{ prime} && (4.2)\\
h_3(u) &= \prod_{\{u,w\}\in E} P_{e(w)} && (4.3)\\
h_2(v) &= \sum_{\{v,u\}\in E} h_3(u) && (4.4)\\
h(G) &= \sum_{v\in V(G)} P_{e(v)}^{h_2(v)} \bmod M && (4.5)
\end{aligned}
$$

Here, M denotes the size of the hash table and is taken to be some appropriately large prime. This hash function encodes the structure of the graph by encoding the number of neighbours that every vertex has, and the number of neighbours of the neighbours. Although some time was spent on devising this hash function, it was only improved until the algorithm performed sufficiently fast. There are certainly better or more efficient ways of generating hash values.

The graph isomorphism check is executed by sorting the vertices by degree. At this point it may already be possible to determine that two graphs are not isomorphic. Otherwise, a recursive bruteforce algorithm will try all possible assignments of vertices with equal degree. Some obvious ways of pruning the search were used to improve performance, but they are of no interest for this thesis.

The method just described clearly has several drawbacks: for example, all graphs have to be stored in memory. One way around this problem is to limit the size of the hash table and accept the fact that some isomorphic graphs will occur in the lists multiple times. Another solution consists of running the algorithm several times for the same n, each time only storing and outputting the graphs on $n+1$ vertices that hash into a particular range. Under the assumption that the distribution of the graphs is roughly uniform amongst the partitions of the hash table, this allows reducing the amount of space required by the algorithm by a factor of k in exchange for an increase of the runtime by a factor of k.

4.1.2 Decreasing Space Complexity

It is possible, in fact, to reduce the amount of space required to $\mathcal{O}(n)$ without increasing the runtime complexity. Note that this space requirement is practically constant, given that we are only interested in graphs of size at most 31. When I had the honour of explaining the topic of my dissertation and my approaches to Prof. D. Knuth, he pointed me to a paper by C. J. Colbourn and R. C. Read on "Orderly Algorithms for Graph Generation" [3]. This very accessible paper describes a method for generating all non-isomorphic graphs with a 'hereditary' property without the need for isomorphism checks. By a 'hereditary' property $P(G)$ we mean a graph property which is satisfied by any subgraph of a graph satisfying P. The term 'hereditary' is not used by Colbourn and Read, but it is clear that their method works for enumerating all graphs with such a hereditary property.

The key idea is that the adjacency matrices of the graphs are generated in unique canonical forms. The canonical form has the property that removing the last row and column of a canonical matrix (i.e. removing the 'last' vertex of the graph) results in another canonical matrix of a graph with one fewer vertex.

When generating the canonical adjacency matrices of graphs on $n+1$ vertices from the canonical $n \times n$ matrices, we add the last column and row and output exactly those adjacency matrices that are already in canonical form (without permuting rows or columns). It is then clear that there is exactly one way of generating the canonical form of any particular graph on $n+1$ vertices.

More concretely, the canonical representation of an undirected simple graph G is taken

to be the following: given the adjacency matrix $\mathbf{M}$ of G, consider the string of 0s and 1s obtained by concatenating the entries above the diagonal of $\mathbf{M}$ in a column-by-column fashion. We call this the *string representation* of $\mathbf{M}$. The canonical representation of G is the (unique) adjacency matrix of G, which has the greatest string representation in lexicographical order. We call the adjacency matrix of a graph canonical iff it is the canonical representation of the graph that it describes.

Note that the graphs under consideration are actually unlabelled graphs, but an adjacency matrix introduces an order or a labelling on the vertices. We show that removing the 'last' vertex with respect to this order results in another canonical adjacency matrix.

Proposition 8. *The top-left* $(n-1) \times (n-1)$ *submatrix of a canonical* $n \times n$ *matrix is canonical.*

Proof. Let $\mathbf{A}$ be the canonical $n \times n$ matrix representing graph G, and $\mathbf{B}$ its $(n-1) \times (n-1)$ top-left submatrix. Note that $\mathbf{B}$ is the adjacency matrix of an undirected simple graph H: it corresponds to the deletion of a single vertex of G.

Let a and b be the string representations of $\mathbf{A}$ and $\mathbf{B}$ respectively. Note that b is just a with the last $n-1$ symbols deleted.

Now suppose, for a contradiction, that $\mathbf{B}$ is not canonical. Let $\mathbf{B}'$ be the canonical representation of H. $\mathbf{B}'$ may be obtained from $\mathbf{B}$ by some permutation of the rows and columns, i.e. by a 're-labelling' of the vertices of H. Re-labelling the vertices of G in the same fashion, we arrive at an adjacency matrix $\mathbf{A}'$ with $\mathbf{B}'$ as its upper-left $(n-1) \times (n-1)$ submatrix. Again, we write a' and b' for the string representations of A' and B'. Note that b' is again just a' with the last $n-1$ symbols deleted. Since $b' > b$ lexicographically, it follows that $a' > a$ lexicographically. This contradicts the assumption that $\mathbf{A}$ is canonical. We deduce that $\mathbf{B}$ must indeed be a canonical adjacency matrix. □

We could have replaced the lexicographical order with any total order on strings of 0s and 1s, so long as for any two strings x and x' of the same length and any string y it holds that $x > x' \implies xy > x'y$. In particular, we could have chosen to use the representation that is *least* in lexicographical order as the canonical representation.

However, being connected and square-free is *not* a hereditary property: a subgraph of

a square-free, connected graph need not be connected. We demonstrate how this could be a problem using a simple example. Suppose we had chosen the matrix with the least string representation as the canonical form.

Starting from the one-vertex graph with representation (0), we generate one two-vertex graph (a single edge):

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The string representation of this graph is just '1'. In the next step, we would like to generate the path on three vertices P_3 and the triangle. However, the canonical matrix representation of P_3 is

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

with string representation '011'. This graph cannot be generated from the single edge, because the canonical representation of the edge is not the top-left 2×2 sub-matrix of the canonical representation of P_3.

Fortunately, we can show that the 'last' vertex in the canonical representation (the greatest string representation) of a simple graph G is never a cut vertex in G.

Proposition 9. *The last vertex in the canonical representation of a connected, simple, undirected graph is not a cut vertex.*

Before we turn to the proof of this result, we need the following auxiliary result.

Lemma 7. *If a simple, undirected graph has multiple components, then all the vertices of one component will appear first in the vertex order imposed by the canonical adjacency matrix.*

Proof. Let $G = (V, E)$ be a simple, undirected graph with at least two components. Suppose $\mathbf{M}$ is the canonical adjacency matrix of G. As remarked previously, $\mathbf{M}$ defines an order on the vertices of G. Let $A \subset G$ be the component of the first vertex in this order. Let $y \in V$ be the first vertex in the order that is not in A. Finally, suppose, for a contradiction, that $z \in V$ is a vertex connected to some vertex that occurs before y in the order, but z itself

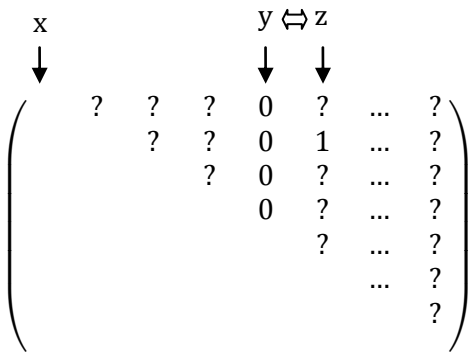

Figure 4.2: Swapping vertices y and z results in a matrix with lexicographically greater string representation.

occurs after y in the order. Note that such a z always exists unless all vertices of component A occur before y in the vertex order. We show that by swapping the order of y and z, one can arrive at an adjacency matrix with lexicographically greater string representation than $\mathbf{M}$. This contradicts the assumption that $\mathbf{M}$ is canonical. Thus, if $\mathbf{M}$ is canonical, then all the vertices of one component appear first in the order imposed by $\mathbf{M}$, as claimed.

Indeed, since y is in a different component than all the vertices occurring before y in the vertex order, all column entries of y above the diagonal must be 0. Now z is connected to some vertex that occurs before y in the vertex order. Swapping z and y will leave the upper-diagonal entries of the columns before the column of y unchanged. The upper-diagonal entries of column y, which are all 0, will be replaced by the corresponding entries of z. At least one of those entries is 1. Thus, the string representation of the matrix obtained by swapping y and z is lexicographically strictly greater than the string representation of $\mathbf{M}$ (c.f. the illustration in Fig. 4.2). This concludes the proof.

□

Proof. (Proposition 9) Let x be the last vertex in the canonical representation of a connected, simple, undirected graph G with adjacency matrix $\mathbf{M}$. We show that if removing x disconnects G, then $\mathbf{M}$ cannot be canonical.

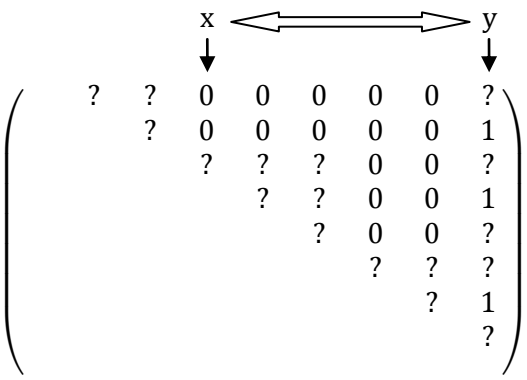

Figure 4.3: Swapping vertices x and y results in an adjacency matrix with lexicographically greater string representation.

Indeed, if x disconnects G, then $G - \{x\}$ has $k > 1$ components. Also, there is at least one edge in G from x to each of the k components of $G - \{x\}$ (because G is connected). From Prop. 8 we know that the canonical adjacency matrix of $G - \{x\}$ is just $\mathbf{M}$ with the last row and column removed. By Lemma 7 all the vertices of a single component $A \subset G$ appear first in the order induced by $\mathbf{M}$. Also, the first vertex y not in A has all entries above the diagonal of $\mathbf{M}$ set to 0. Swapping x and y will leave all entries above the diagonal to the left of y's column unchanged. However, at least one of the entries above the diagonal in y's column will become 1. This contradicts the assumption that $\mathbf{M}$ is canonical. Therefore, if $\mathbf{M}$ is the canonical matrix of a connected, simple, undirected graph, then the last vertex in the vertex order induced by $\mathbf{M}$ is not a cut vertex. This idea is illustrated in Fig. 4.3.

□

Note that this result allows us to enumerate all *connected* graphs with any hereditary property. In particular, we can enumerate all connected square-free graphs. This result is, in some sense, an extension of the method suggested by Colbourn and Read.

We can now simply enumerate all connected square-free graphs with at most 16 vertices and check each of them for 101-colourability. This allows us to obtain the following result.

Theorem 5. *Every square-free graph with at most 16 vertices is 101-colourable.*

Proof. (*computer-aided*) Note that it suffices to show the result for all connected square-free graphs: A graph consisting of several square-free components is 101-colourable iff every one of its component is 101-colourable. Using the methods described above, all connected square-free graphs with up to 16 vertices were enumerated and checked for 101-colourability. □

4.1.3 Finding Small Uncolourable Square-Free Graphs

In this section we describe the method employed to find a graph on 17 vertices that is square-free but not 101-colourable. From the previous section we already know that no graph on fewer vertices with this property exists.

The basic idea is that we generate a 'large' square-free graph that is not 101-colourable on N vertices ($N = 35$, say) at random. Once we have such a graph, a backtracking algorithm is used to find the smallest uncolourable subgraph.

To generate the large graph on N vertices, we take a permutation of all $\binom{N}{2}$ potential edges uniformly at random. This list of edges is then processed from the beginning to the end. At every step, the current edge is added to the graph if it does not introduce a square. Once the complete list is processed, the resulting graph is checked for 101-colourability. If it is not 101-colourable, then the smallest uncolourable subgraph is extracted.

It turns out that many of the graphs generated in this way are uncolourable. This observation has not been analysed, but it appears reasonable to make the following conjecture: In the limit, as $N \to \infty$, the probability that a randomly generated, maximal, square-free graph on N vertices is 101-colourable tends to 0 or some other constant less than 1. By 'randomly generated', we mean generated by the method described in the previous paragraph. It is possible that the statement also holds if the graphs are generated uniformly at random.

Using the method just described, it was possible to generate several hundred 17 vertex graphs within about a day. All of the graphs were isomorphic to the graph displayed in Figure 4.4. Generating nice (symmetric) plots of graphs is a difficult problem. The plot shown is the most symmetric one found, but there may well be plots that reveal more interesting aspects of the structure. For completeness, we give the list of edges of the

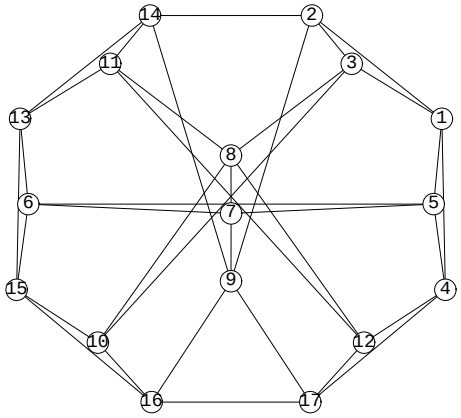

Figure 4.4: A graph on 17 vertices that is not 101-colourable.

graph:

$$
\begin{aligned}
V =& \{1, \ldots, 17\} \\
E =& \left\{ \begin{array}{l}
\{1,2\},\{1,9\},\{1,10\},\{1,15\},\{2,3\},\{2,8\},\{2,9\}, \\
\{3,4\},\{3,8\},\{3,11\},\{4,12\},\{4,15\},\{4,16\},\{4,17\}, \\
\{5,6\},\{5,7\},\{5,8\},\{5,12\},\{6,10\},\{6,11\},\{6,12\}, \\
\{7,8\},\{7,13\},\{7,16\},\{9,14\},\{9,17\},\{10,11\},\{10,15\} \\
\{11,13\},\{11,14\},\{12,17\},\{13,14\},\{13,16\},\{14,17\},\{15,16\}
\end{array} \right\}
\end{aligned}
$$

The existence of this graph naturally leads to the following two questions: Does this graph have a corresponding direction set? Is this the only graph on 17 vertices that is not 101-colourable? Unfortunately, the answer to the first question is 'No'. However, the graph is indeed the unique square-free graph on 17 vertices that is not 101-colourable. Those two facts together allow us to establish a new lower bound for the smallest KS vector system: A three-dimensional KS vector system must contain at least 18 vectors. We now turn to the proofs of these facts.

4.1.4 Uniqueness and Non-Embeddability of the 17 Vertex Graph

Proposition 10. *The square-free graph shown in Fig. 4.4 is the unique graph with 17 vertices that is not 101-colourable.*

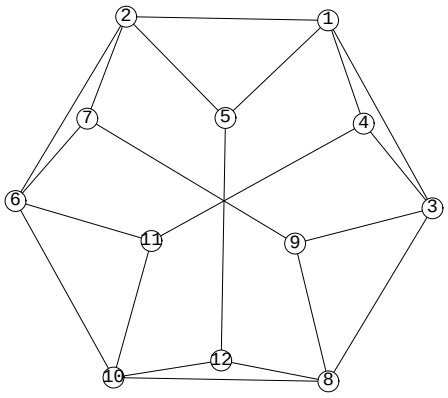

Figure 4.5: An unembeddable subgraph of the uncolourable graph on 17 vertices.

Proof. (*computer-aided*) By enumeration of all square-free graphs with 17 vertices. □

In some sense, all of the methods introduced in Chapter 3 play a role in demonstrating that this graph has no embedding. First, Mathematica was used to extract the smallest subgraph whose embedding polynomial could not be minimized to (approximately) 0 using `NMinimize`.

The graph that was found is shown in Fig. 4.5. Of course, the fact that Mathematica cannot find a real root of the embedding polynomial for this graph does not mean that this graph is not embeddable. Nonetheless, the graph serves as a good starting point.

It is a very fortunate coincidence that this graph has exactly the right number of edges: 12 vertices mean 24 degrees of freedom. Three degrees of freedom are removed by fixing the location of the vectors corresponding to the vertices of a triangle. Thus, 21 degrees of freedom are left – exactly one degree of freedom per edge. Thanks to the help of Prof. Charles W. Wampler, who created the Bertini input file shown in Sect. 7.2 of the Appendix, we were able to show the following:

Proposition 11. *The 12 vertex graph shown in Fig. 4.5 is not embeddable.*

Proof. (*numerical*) Bertini finds 12 non-singular roots (and no other roots), which are all complex. □

Since the orthogonality equations admit no real solutions, we know by Prop. 7 that the

graph shown in Fig. 4.5 cannot be 3-colourable. Indeed, we can show the following result.

Proposition 12. *The 12 vertex graph shown in Fig. 4.5 is the unique smallest square-free graph that is not 3-colourable.*

Proof. (*computer-aided*) By enumeration and checking of the connected square-free graphs on up to 12 vertices. □

Corollary 3. *The orthogonality equations of any square-free graph with fewer than 12 vertices admit at least one real solution.*

Proof. From Prop. 7 and Prop. 12. □

We can see that it is no coincidence that this graph was found to be the smallest subgraph of the 17 vertex graph for which `NMinimize` does not find a global minimum that is very close to 0. Every smaller subgraph admits at least the real solutions that are induced by its three-colourings, as described in the proof of Prop. 7.

Since none of the solutions are real, we can turn the numerical proof of Prop. 11 into a computer-aided proof.

Proof. (of Prop. 11, *computer-aided*) Using interval analysis, without discarding solutions that have vectors that lie too close to each other. □

Prop. 11 allows us to prove our final theorem regarding the lower bound.

Theorem 6. *A Kochen-Specker vector system must contain at least 18 directions.*

Proof. (*computer-aided*) All square-free graphs with fewer than 17 vertices are 101-colourable (Thm. 5). The unique uncolourable graph on 17 vertices is not embeddable (Prop. 10 and Prop. 11). □

4.1.5 Generating Embeddable Connected Graphs

Given the large number of square-free graphs on 16 vertices, one might ask how many of such graphs are actually embeddable. It is reasonable to assume that the number of graphs under consideration can be decreased by filtering out the graphs that have no embeddings.

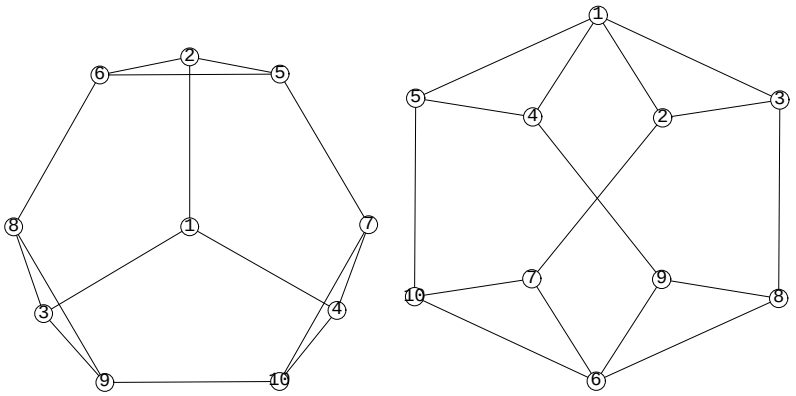

Figure 4.6: The only two square-free graphs on 10 vertices for which no embedding was found.

A program was written to find the smallest square-free graph that has no embedding on the box grid with grid parameter $N = 4$. The following result may be surprising:

Proposition 13. *Every square-free graph with fewer than 10 vertices is embeddable on the box grid with grid parameter $N = 4$.*

Proof. (*computer-aided*) By enumeration of all square-free graphs with fewer than 10 vertices and finding embeddings for all of them. □

This result indicates that the square-freeness is already a strong condition for a graph to be embeddable. There are two graphs on 10 vertices that were (numerically) found not to be embeddable. All other square-free graphs on 10 vertices are embeddable.

Proposition 14. *The two graphs shown in Fig. 4.6 are not embeddable.*

Proof. (*numerical*) Using interval analysis. The proof is numerical, because the possibility that two directions in an embedding make a tiny angle cannot be excluded. Such an embedding would be rejected by the interval analysis embedder. □

Out of the the 18502 square-free connected graphs with 11 vertices, 18484 were found to be embeddable. One can see that removing unembeddable graphs may not reduce the number of graphs under consideration as much as hoped. We can certainly always just add one single edge connecting a new vertex to a smaller embeddable graph to arrive at

another embeddable graph. Unless the small graph is completely symmetric, there are many possibilities of connecting an additional vertex using one edge. Therefore, the number of embeddable graphs can intuitively be expected to increase at least exponentially with the number of vertices.

Once a new graph is generated, one can check whether it contains a subgraph isomorphic to an unembeddable graph. As the number of known unembeddable graphs increases, this check may become more expensive than running the embeddability check again. Both methods increase the runtime of the algorithm drastically, unless the embeddability check is very efficient. The automated embedding algorithm that was used already had problems with some of the 12 vertex graphs.

Chapter 5

Upper Bound

The current upper bound for the size of the smallest KS system in three dimensions is Conway's system (see Fig. 5.1; the construction is taken from [19]). In chapter 7.3 we give the coordinates of the system, as well as its graph representation.

5.1 Generating Uncolourable Graphs at Random

Some attempts were made to find smaller systems, but none of them were successful. The biggest problem encountered lies in checking whether or not a graph is embeddable. In fact, the algorithm outlined in 4.1.3 finds many square-free graphs that are not 101-colourable

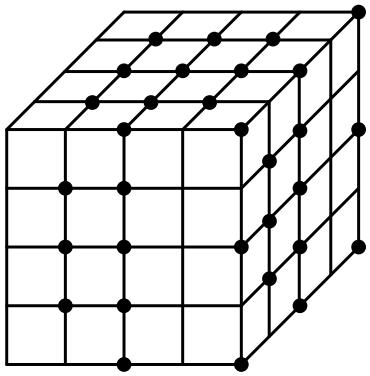

Figure 5.1: Construction of Conway's KS vector system with 31 directions.

and have fewer than 31 vertices. Unfortunately, no embedding for any of the graphs was found (this does not mean that no embedding exists, of course). If the embeddability of a graph could be decided efficiently, then it would be interesting to use the algorithm to generate embeddable, uncolourable graphs at random (although probably far from uniformly). Looking at the smallest uncolourable subgraphs, one might find a smaller KS vector system, or provide some evidence that Conway's KS vector system is indeed the smallest one in three dimensions.

5.2 Uncolourable Subgraphs of the Box Grid

Another method of finding upper bounds is to take a large, embeddable graph which is known to be not 101-colourable and to find its smallest uncolourable subgraph. We call a smallest such subgraph an *uncolourable core*. By smallest, we mean with the smallest number of vertices. We are not interested in the number of edges. A graph that has no proper subgraph (no subgraph with fewer vertices) is called *minimal*.

For example, the graph representation of Conway's KS vector system is a minimal uncolourable subgraph of the graph representation of the box grid with grid parameter $N = 2$. No efficient algorithm that produces the smallest uncolourable subgraph of a given graph was found. However, the following observations help pruning the backtracking search.

Proposition 15. *A minimal not 101-colourable graph must have minimum degree at least 3.*

Proof. By definition, any proper subgraph (i.e. subgraph with fewer vertices) of a minimal graph is 101-colourable. We show that if we add a vertex to a colourable graph and connect it to at most two other vertices, then this new graph can still be 101-coloured. Indeed, if the new vertex has fewer than two neighbours, then it cannot be the member of any triangle. Thus, we can certainly assign 1 to the new vertex and colour the rest of the graph as before.

The remaining configurations are shown in Fig. 5.2. The newly added vertex is the vertex on the left while the two neighbours are the two vertices on the right. Only one configuration is shown in case there are multiple symmetric configurations. Note that we

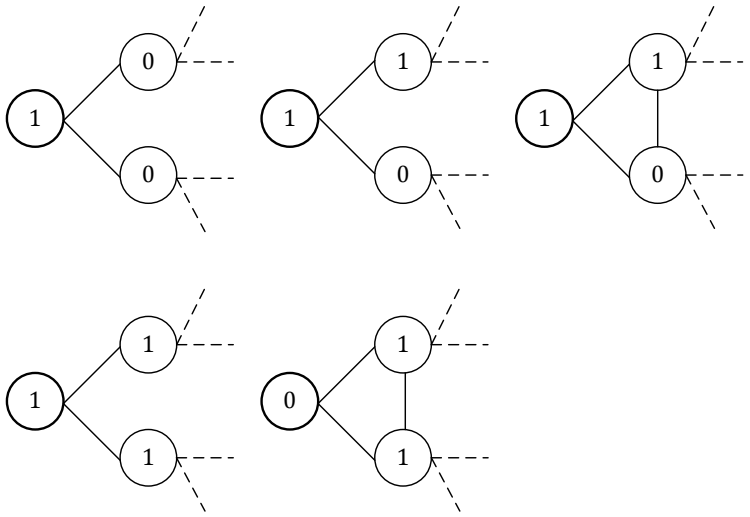

Figure 5.2: Possible configurations when connecting a new vertex to two neighbours.

only need to check the colourability conditions locally, i.e. within at most one triangle that the new vertex belongs to, and for the two edges. The conditions are satisfied on the remaining edges and triangles, because the original graph is 101-colourable. □

Remark 5. *As the graph representation of Conway's KS vector system shows, it is possible to have minimum degree 3 in a minimal uncolourable graph.*

Proposition 16. *Every vertex in a minimal not 101-colourable graph must belong to at least one triangle.*

Proof. Suppose, for a contradiction, that there exists a minimal graph $G = (V, E)$ that is not 101-colourable and which contains a vertex x that does not belong to a triangle. By minimality, removing the vertex x results in a 101-colourable graph $G - \{x\}$. Let $c : V - \{x\} \mapsto \{0, 1\}$ be a 101-colouring of $G - \{x\}$. Then $c' : V \mapsto \{0, 1\}$ defined by

$$c'(v) = \begin{cases} c(v) & \text{if } v \neq x \\ 1 & \text{if } v = x \end{cases}$$

is a valid 101-colouring of G, because x does not belong to any triangle and can therefore be coloured 1 in any case. □

Finding an uncolourable core of a large graph is infeasibly slow. However, the following result was obtained.

Proposition 17. *There is no smaller subgraph of the box grid graph with parameter $N = 2$ than Conway's graph.*

Proof. (*computer-aided*) By enumeration of the uncolourable cores using a backtracking algorithm. □

For larger values of N, the algorithm was too slow to generate all minimal uncolourable subgraphs. Instead, the following algorithm was used to look for smaller uncolourable subgraphs: We maintain a subgraph H of the box grid graph. At each step, a neighbour of H which is not already contained in H is added at random (the probability being proportional to how many edges of the vertex cross the cut between H and $G - H$). At some point, H becomes uncolourable (note that this means that the vertex added in the last step definitely belongs to the smallest uncolourable subgraph of H). The algorithm then spends some amount of time looking for the smallest uncolourable subgraph of H.

Using this method, the smallest subgraphs found with box grid parameters $N = 6$, 8, or 12 all contained 31 vertices. For example, 44 uncolourable subgraphs on 31 vertices were discovered. All of them were isomorphic. This supports the idea that the box grid with parameter $N = 8$ contains no smaller KS vector system than Conway's.

For odd box grid parameters, the smallest box grid graph that is not 101 colourable has $N = 15$. Not much time was spent investigating the uncolourable subgraphs.

Chapter 6

Discussion

6.1 Results

The results obtained in this thesis can almost completely be summarised by giving a characterisation of the smallest Kochen-Specker vector system in three dimensions. The smallest such system:

- has at most 31 directions (Conway's KS vector system)
- has at least 18 directions (Thm. 6)
- has graph representation G that is square-free (Prop. 2)
- has graph representation G that is not 3-colourable (Prop. 1)
- has graph representation G that is 4-colourable (Prop. 3)
- has graph representation G with minimum degree 3 (Prop. 15)
- has graph representation G that is not isomorphic to a subgraph of the graph representation of the box grid with grid parameter $N = 2$ (Prop. 17)
- has graph representation G in which each vertex belongs to a triangle (Prop. 16)

The thesis also contains some minimality results (Prop. 4, Prop. 10, Prop. 12, Prop. 17) and an explanation for the large number of real solutions to the orthogonality equations

of some graphs (Prop. 7). Furthermore, it was shown that checking 101-colourability of (square-free) graphs is NP-complete (Thm. 2 and Thm. 3), although the proof idea is due to Dr Joël Ouaknine. It was also shown that the method of orderly graph generation described by Colbourn and Reed [3] works for the generation of all connected graphs with a descending property.

It is natural to ask what proportion of square-free, connected graphs with up to and including 30 vertices were covered in our search for a lower bound. We know that the total number of labelled graphs on n vertices is just $2^{\binom{n}{2}}$. We can use the numbers of square-free, connected graphs on up to 17 vertices in order to fit a function of the form e^{ax^2+bx+c} to the data. Using Microsoft Excel, an estimate of $10^{0.0349x^2+0.0059x}$ was determined (c.f. Fig. 6.1). Note that we have no reason to claim or assume that this is a reasonable estimate. Extrapolating this function, we find that there should be approximately 10^{32} square-free connected graphs with 30 vertices. A total of $10^{10.28}$ graphs were scanned, i.e. only about 10^{-22} of our estimate of the work space. This indicates that the method of enumerating all candidates may currently be infeasible. Given an efficient embeddability tester, the randomised algorithm suggested in this thesis may at least provide an indication of what the smallest number of directions required in a three-dimensional Kochen-Specker vector system is.

6.2 Failed Attempts

Naturally, there was a large number of approaches and attempts which failed and did not even make it into the write-up. Many of them are not worth mentioning, but some of them probably deserve some attention.

One such approach is the following: one could conjecture that graphs are often not embeddable if they have more embedding equations than degrees of freedom. Simply spoken, one could require that the maximum density subgraph must have a density of at most 2: on average, at most two equations limit the two degrees of freedom of each vertex. An algorithm for finding the maximum density subgraph [10] was implemented, but as it turns out, the graph representation of Conway's KS vector system already has a maximum density subgraph with density $\frac{12}{5}$. This shows that discarding graphs with high density subgraphs

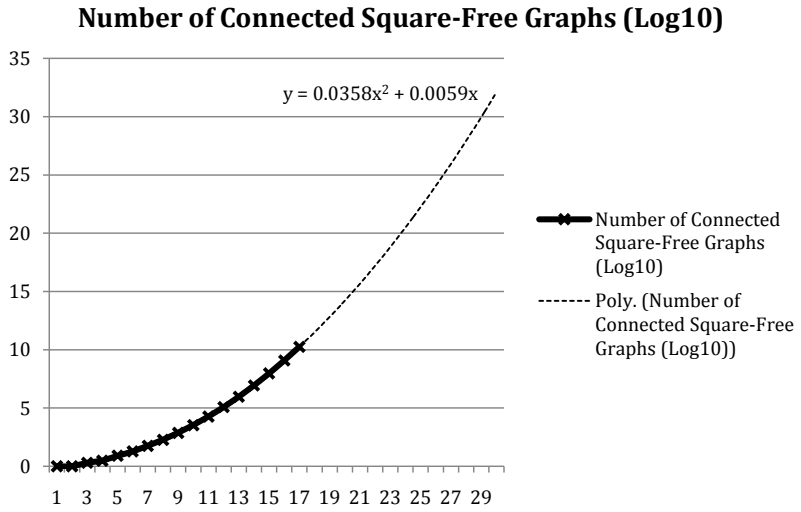

Figure 6.1: An estimate of the number of square-free, connected graphs with more vertices.

is probably not a good idea.

Several attempts involving Ordered Binary Decision Diagrams (OBDDs) were made. For example, such diagrams may be used to count the number of 101-colourings of a given graph. Using this number in a heuristic to find the smallest uncolourable subgraph of a given graph resulted in no improvement over the obvious backtracking algorithm. It was also investigated whether OBDDs could be used to count the number of square-free graphs on n vertices (counting isomorphic graphs multiple times), but the size of the OBDDs becomes unmanageable very quickly. Intuitively, this is clear: the ordering of the variables (the edges) cannot have a huge influence of the size of the OBBDs, because the problem is symmetric for all edges. Whether or not a particular edge is present influences the presence of squares – also squares that contain edges that occur at the end of the variable order. This means that the OBDD somehow has to represent the states of (almost) all edges and not too many simplifications can be made.

6.3 Future Work

The problem of determining the number of necessary directions in a Kochen-Specker vector system remains unsolved. In my opinion, the biggest open problem is determining efficiently whether or not a given graph has an embedding. Some of the difficulties certainly result from the potentially large number of real solutions (with collinear vectors) of 3-colourable graphs.

Another area which might be interesting to investigate is whether or not every finite graph has an embedding with only rational coordinates on the surface of the unit cube $[-1, 1]^3$ (Conjecture 1). If one could also establish a reasonable upper bound for the grid parameter of the grid required to embed a graph of given size, then finding the truly smallest KS vector system in three dimensions would be reduced to finding the smallest uncolourable cores of a few graphs.

Finally, I believe that it is not well enough understood which property exactly makes a graph unembeddable. The square-freeness of a graph is a very simple property, and it eliminates all unembeddable graphs with up to and including 9 vertices. It would be nice to find more such properties, and to devise an algorithm that can generate graphs which satisfy those properties almost uniformly at random. I have not been able to obtain any of my lower bound results purely analytically – computer programs where involved in all the proofs. Analytic proofs of (possibly slightly weaker) results would probably provide a lot of insight as to why so many directions are required to establish a contradiction from the SPIN axiom.

Chapter 7

Appendix

7.1 Numerical Minimization Example

We show how to find an embedding for Conway's KS vector system using numerical minimzation.

```
v[i_]:={x[i],y[i],z[i]};
f:=Sum[(1-v[i].v[i])^2,{i,0,30}]+
  (v[0].v[11])^2+(v[0].v[13])^2+(v[0].v[20])^2+
  (v[0].v[29])^2+(v[1].v[15])^2+(v[1].v[19])^2+
  (v[1].v[20])^2+(v[2].v[12])^2+(v[2].v[13])^2+
  (v[2].v[22])^2+(v[3].v[17])^2+(v[3].v[20])^2+
  (v[3].v[26])^2+(v[3].v[28])^2+(v[4].v[14])^2+
  (v[4].v[15])^2+(v[4].v[21])^2+(v[4].v[22])^2+
  (v[4].v[26])^2+(v[4].v[29])^2+(v[5].v[13])^2+
  (v[5].v[24])^2+(v[5].v[26])^2+(v[5].v[30])^2+
  (v[6].v[16])^2+(v[6].v[17])^2+(v[6].v[22])^2+
  (v[7].v[15])^2+(v[7].v[23])^2+(v[7].v[24])^2+
  (v[8].v[17])^2+(v[8].v[24])^2+(v[8].v[27])^2+
  (v[8].v[29])^2+(v[9].v[18])^2+(v[9].v[22])^2+
  (v[9].v[23])^2+(v[9].v[27])^2+(v[9].v[28])^2+
  (v[10].v[12])^2+(v[10].v[15])^2+(v[10].v[25])^2+
  (v[10].v[27])^2+(v[10].v[30])^2+(v[11].v[16])^2+
  (v[11].v[18])^2+(v[11].v[25])^2+(v[11].v[29])^2+
  (v[12].v[21])^2+(v[12].v[27])^2+(v[13].v[22])^2+
```

```
  (v[14].v[19])^2+(v[14].v[21])^2+(v[14].v[23])^2+
  (v[15].v[20])^2+(v[15].v[22])^2+(v[15].v[24])^2+
  (v[15].v[25])^2+(v[16].v[21])^2+(v[16].v[25])^2+
  (v[17].v[22])^2+(v[18].v[19])^2+(v[18].v[22])^2+
  (v[18].v[30])^2+(v[19].v[30])^2+(v[23].v[28])^2+
  (v[25].v[28])^2+(v[26].v[28])^2+(v[26].v[29])^2+
  (v[26].v[30])^2+(v[27].v[29])^2;
soln=NMinimize[f,
    Union[v[0],v[1],v[2],v[3],v[4],v[5],v[6],v[7],
          v[8],v[9],v[10],v[11],v[12],v[13],v[14],
          v[15],v[16],v[17],v[18],v[19],v[20],v[21],
          v[22],v[23],v[24],v[25],v[26],v[27],v[28],
          v[29],v[30]]]
```

Using Mathematica 5.2, the following minimum was found:

Vertex	x	y	z
1	0.83434	-0.353089	0.423325
2	0.643305	-0.170387	0.746409
3	0.597761	-0.696065	0.397714
4	-0.340168	-0.042007	-0.939426
5	0.365699	-0.536278	0.760704
6	-0.257016	0.917745	-0.302798
7	0.0564223	-0.263258	0.963074
8	-0.0108778	0.788936	-0.614379
9	-0.237156	-0.522649	0.818899
10	-0.169377	-0.0370425	0.984855
11	-0.241389	-0.868212	0.433519
12	0.546526	0.429517	-0.718905
13	-0.0562136	0.45849	0.88692
14	-0.377792	0.192975	0.905557
15	0.225125	-0.742081	-0.631375
16	0.605234	-0.483892	-0.632092
17	0.354806	-0.896359	-0.265808

18	0.704884	-0.672637	-0.225163
19	0.686554	-0.721369	0.0909424
20	-0.675114	-0.586042	0.448079
21	-0.468882	-0.85838	0.208166
22	0.903096	0.402147	-0.150649
23	0.707075	0.69156	0.147615
24	0.973706	0.148173	0.173033
25	-0.795973	-0.378719	-0.472228
26	0.758566	0.109801	0.642278
27	0.927943	0.146843	-0.342577
28	0.968799	-0.189723	0.15948
29	-0.152339	0.988267	0.0109713
30	0.0720122	0.83117	0.551336
31	-0.269934	-0.369027	-0.889356

The function value at these points is approximately $2.74 \cdot 10^{-31}$. The smallest length of the cross-product between any two distinct vectors is approximately 0.1.

7.2 A Bertini Input File with Isolated Solutions

```
% A graph on 12 vertices with 21 edges.
% If all 21 eqns are independent, we have 2 * 12 - 21 = 3
% dimensional solution but we can take out 3 dimensions by
% modding out by O(3).
% This is done by fixing the coordinates for one triangle.
% Vertex 0 fixed at (1, 0, 0).
% Vertex 1 fixed at (0, 1, 0).
% Vertex 7 fixed at (0, 0, 1).
CONFIG
 MPTYPE: 2;
 DEGREEBOUND: 2;
```

```
 COEFFBOUND: 1;
END
INPUT
 hom_variable_group y2, z2;
 hom_variable_group x3, z3;
 hom_variable_group x4, y4, z4;
 hom_variable_group x5, y5, z5;
 hom_variable_group y6, z6;
 hom_variable_group x8, z8;
 hom_variable_group x9, y9, z9;
 hom_variable_group x10, y10, z10;
 hom_variable_group x11, y11;

 function e2e4, e2e6, e2e9, e3e5, e3e8, e3e10, e4e5,
          e4e9, e4e11, e5e10, e5e11, e6e10, e8e9;

 e2e4 =  y2 * y4 + z2 * z4;
 e2e6 =  y2 * y6 + z2 * z6;
 e2e9 =  y2 * y9 + z2 * z9;
 e3e5 =  x3 * x5 + z3 * z5;
 e3e8 =  x3 * x8 + z3 * z8;
 e3e10 =  x3 * x10 + z3 * z10;
 e4e5 =  x4 * x5 + y4 * y5 + z4 * z5;
 e4e9 =  x4 * x9 + y4 * y9 + z4 * z9;
 e4e11 =  x4 * x11 + y4 * y11;
 e5e10 =  x5 * x10 + y5 * y10 + z5 * z10;
 e5e11 =  x5 * x11 + y5 * y11;
 e6e10 =  y6 * y10 + z6 * z10;
 e8e9 =  x8 * x9 + z8 * z9;
END
```

All 12 solutions are complex.

7.3 Representations of Conway's KS vector system

The following table lists the coordinates of the vectors of Conway's KS vector system. The vectors lie on the surface of the cube $[-2, 2]^3$.

1:	(-1, 2, 1)	9:	(1, 2, -1)	17:	(-1, -1, -2)	25:	(2, -1, 0)
2:	(-1, 2, 0)	10:	(0, 2, -2)	18:	(0, -1, -2)	26:	(2, -2, 0)
3:	(0, 2, 1)	11:	(2, 2, 0)	19:	(0, -2, -2)	27:	(2, 0, -2)
4:	(-1, 2, -1)	12:	(2, 2, -2)	20:	(2, 1, -1)	28:	(2, -2, -2)
5:	(0, 2, 0)	13:	(-1, 1, -2)	21:	(2, 1, 0)	29:	(2, 2, 2)
6:	(1, 2, 1)	14:	(0, 1, -2)	22:	(2, 0, -1)	30:	(2, 0, 2)
7:	(0, 2, -1)	15:	(-1, 0, -2)	23:	(2, 0, 0)	31:	(2, -2, 2)
8:	(1, 2, 0)	16:	(0, 0, -2)	24:	(2, -1, -1)		

The graph representation can be given as an adjacency list. Line i contains the degree d_i of vertex i in the graph. The degree is followed by a list of the d_i neighbours of vertex i. The vertices are numbered from 1 to 31.

```
31
4 12 14 21 30
3 16 20 21
3 13 14 23
4 18 21 27 29
6 15 16 22 23 27 30
4 14 25 27 31
3 17 18 23
3 16 24 25
4 18 25 28 30
5 19 23 24 28 29
5 13 16 26 28 31
5 1 17 19 26 30
4 3 11 22 28
4 1 3 6 23
4 5 20 22 24
8 2 5 8 11 21 23 25 26
4 7 12 22 26
4 4 7 9 23
5 10 12 20 23 31
4 2 15 19 31
4 1 2 4 16
4 5 13 15 17
8 3 5 7 10 14 16 18 19
```

```
4 8 10 15 29
4 6 8 9 16
5 11 12 16 17 29
6 4 5 6 29 30 31
5 9 10 11 13 30
5 4 10 24 26 27
6 1 5 9 12 27 28
5 6 11 19 20 27
```

The graph contains 17 triangles and 71 edges.

Bibliography

[1] D. J. Bates, J. D. Hauenstein, J. Sommese, and C. W. Wampler. Bertini home page. `http://www.nd.edu/~sommese/bertini/`.

[2] John Canny. Some algebraic and geometric computations in pspace. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 460–469, New York, NY, USA, 1988. ACM.

[3] C. J. Colbourn and R. C. Read. Orderly algorithms for graph generation. *International Journal of Computer Mathematics*, 7:167–172, 1979.

[4] John Conway and Simon Kochen. The free will theorem, 2006.

[5] John Conway and Simon Kochen. The strong free will theorem, 2008.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, September 2001.

[7] N. Eén and N. Sörensson. Minisat. `http://www.minisat.se/`.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W. H. Freeman, January 1979.

[9] C D Godsil and J Zaks. Coloring the sphere. university of waterloo research report corr, 1988.

[10] A. V. Goldberg. Finding a maximum density subgraph. Technical report, Berkeley, CA, USA, 1984.

[11] A W Hales and E G Straus. Projective colorings. *Pac. J. Math*, (99):31–43, 1982.

[12] Yun-Feng Huang, Chuan-Feng Li, Yong-Sheng Zhang, Jian-Wei Pan, and Guang-Can Guo. Experimental test of the kochen-specker theorem with single photons. *Phys. Rev. Lett.*, 90(25):250401, Jun 2003.

[13] Simon Kochen and Ernst P Specker. The problem of hidden variables in quantum mechanics. *Journal of Mathematics and Mechanics*, (17):235–263, 1967.

[14] Ye Lu, Daniel J. Bates, Andrew J. Sommese, and Charles W. Wampler. Finding all real points of a complex curve. Technical report, In Algebra, Geometry and Their Interactions, 2006.

[15] David Meyer. Finite precision measurement nullifies the kochen-specker theorem, 1999.

[16] Mladen Pavičić, Jean-Pierre Merlet, Brendan McKay, and Norman D. Megill. Kochen-specker vectors. *MATHEMATICAL AND GENERAL*, 38:1577, 2005.

[17] Mladen Pavičić, Jean-Pierre Merlet, and Norman Megill. Exhaustive enumeration of Kochen-Specker vector systems. Research Report RR-5388, INRIA, 11 2004.

[18] A Peres. Two simple proofs of the kochen-specker theorem. *J. Phys. A: Math. Gen*, (24):175–178, 1991.

[19] A. Peres. *Quantum Theory: Concepts and Methods*. Kluwer Academic Publishers, 1993.

[20] PROFIL home page. `www.ti3.tu-harburg.de/`.

[21] The COPRIN Project. Alias home page. `http://www-sop.inria.fr/coprin/logiciels/ALIAS/`.

[22] James Renegar. On the computational complexity and geometry of the first-order theory of the reals. parts i-iii. *J. Symb. Comput.*, 13(3):255–352, 1992.

[23] N. J. A. Sloane. The on-line encyclopedia of integer sequences. `http://www.research.att.com/~njas/sequences/`.

[24] A J Sommese and C W Wampler. *The Numerical Solution to Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005.