KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Studying Cycles with Computational Graph Theory

Jarne Renders

Dissertation presented in partial fulfilment
of the requirements for the degree of
Doctor of Engineering Science (PhD):
Computer Science

April 2025

Supervisors:
Prof. dr. J. Goedgebeur
Prof. dr. C.T. Zamfirescu
(Ghent University)

STUDYING CYCLES WITH COMPUTATIONAL GRAPH THEORY

Jarne RENDERS

Supervisors:
Prof. dr. J. Goedgebeur
Prof. dr. C.T. Zamfirescu
(Ghent University)

Members of the
Examination Committee:
Prof. dr. Hendrik Van Brussel, chair
Prof. dr. Hendrik Blockeel
Prof. dr. Greet Vanden Berghe
Prof. dr. Edita Máčajová
(Comenius University)
Prof. dr. Pascal Schweitzer
(TU Darmstadt)

Dissertation presented in partial fulfilment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

April 2025

Uitgegeven in eigen beheer, Jarne Renders, Ghent (Belgium)

Popularised Abstract

Graphs are mathematical objects that store information on whether or not certain items are in relationship with one another. These items are represented by *vertices* and two items are in a relationship if there is an *edge* between the corresponding vertices. This gives rise to a nice graphical representation where one can draw the vertices as dots and draw a line between two dots if there is an edge between them.

Graphs are used to model many things. For example, a road network can be modelled by a graph by letting each intersection be a vertex and drawing an edge between two vertices if there is a road between the intersections. In this way many (unnecessary) details are omitted and one can use this graph model to efficiently determine a shortest path between two vertices. Another example is to model social networks as a graph. Each person is represented by a vertex and two vertices share an edge if the persons are friends. One can then extract interesting information from the network such as whether certain clusters of people appear or how information spreads throughout the network. These are just two examples, but in practice graphs are used to model extremely many things.

The approach we take makes an abstraction of the concrete underlying application(s) and instead looks at the graphs as purely mathematical objects. In this way, we prove statements about graphs that hold true in general. The aim of this is to extend the current knowledge of graph theory, which in turn might contribute to applications.

Most applications of graph theory have in common that things are modelled by a graph in order to compute a certain property (a shortest path, the existence of a cluster, etc.). This is often done via a computer algorithm. In contrast, in theoretical research on graph theory computational methods are not often used, despite the fact that also here, they can be of use. For example if a researcher asks the question whether something holds true for all graphs of a

specific type, one might write a program that checks this for small examples and might immediately find that it does not hold true. Or if very few graphs in an important class are known, one might write a program that generates examples that can be studied in order to gain more intuition.

In this thesis, we apply such computational methods to several topics in structural graph theory and attempt to advance their state-of-the-art research. By doing so, we will show that the combination of computational and theoretical tools can effectively advance the state-of-the-art in this domain.

Gepopulariseerde Samenvatting

Grafen zijn wiskundige objecten die informatie bewaren over het al dan niet in relatie staan van bepaalde zaken. Deze zaken worden voorgesteld door *knopen* en twee zaken staan in relatie als er een *boog* is tussen de bijhorende knopen. Deze voorstelling geeft aanleiding tot een mooie grafische representatie van een graaf waarin de knopen getekend worden als punten en er een lijn is tussen twee punten als er een boog is die de knopen verbindt.

Grafen worden gebruikt voor het modelleren van veel zaken. Bijvoorbeeld een wegennet kan gemodelleerd worden door een graaf waarbij elke knoop een kruispunt voorstelt en twee knopen verbonden zijn als er een weg is tussen de twee kruispunten. Op deze manier worden veel (onnodige) details weggelaten en kunnen we dit grafenmodel gebruiken om op efficiënte wijze een kortste pad te bepalen tussen twee knopen. Een ander voorbeeld is om een sociaal netwerk te modelleren als graaf. Elke persoon wordt voorgesteld door een knoop en twee knopen delen een boog als de personen vrienden zijn. Op deze manier kan interessante informatie afgeleid worden uit het netwerk, zoals waar bepaalde clusters van mensen verschijnen of hoe informatie zich door het netwerk verspreidt. Dit zijn slechts twee voorbeelden, maar in de praktijk worden grafen gebruikt om een enorme verscheidenheid aan zaken te modelleren.

De aanpak die wij volgen maakt abstractie van de concrete onderliggende toepassing(en) en kijkt in plaats daarvan naar grafen als puur wiskundige objecten. Op deze manier bewijzen we stellingen over grafen die waar zijn in het algemeen. Dit heeft als doel de huidige theoretische kennis over grafentheorie te vergroten en deze kennis kan dan op zijn beurt bijdragen aan toepassingen.

De meeste toepassingen van grafentheorie hebben gemeen dat zaken gemodelleerd worden door een graaf om een bepaalde eigenschap (kortste pad, bestaan van een cluster, etc.) te berekenen. Dit gebeurt vaak via een computeralgoritme.

Daarentegen worden computationele methoden vaak niet gebruikt wanneer het aankomt op de theoretische studie van grafentheorie. Dit ondanks het feit dat zulke methoden ook hier nuttig kunnen zijn. Bijvoorbeeld wanneer een onderzoeker vraagt of iets waar is voor alle grafen van een bepaald type, zou men een programma kunnen schrijven dat dit nagaat voor alle kleine voorbeelden en mogelijks onmiddellijk constateren dat dit niet het geval is. Of als slechts een beperkt aantal grafen in een belangrijke klasse bekend zijn, kan men een programma schrijven dat voorbeelden genereert. Deze kunnen dan bestudeerd worden om meer inzicht te krijgen in de klasse.

In deze thesis passen we zulke computationele methoden toe op verschillende topics binnen structurele grafentheorie en proberen op die manier de grenzen van het state-of-the-art onderzoek te verleggen. Door dit te doen zullen we aantonen dat een combinatie van computationele en theoretische methoden effectief is om dit doel te bereiken.

Abstract

In applications of graph theory, computational methods seem to be an inevitability, but despite the growing popularity of computer methods in various areas and the general increase in computing power, such methods are still often underutilised in the domain of structural graph theory.

In this thesis, we aim to push the state-of-the-art structural graph theory research by utilising computational methods alongside a more theoretical approach. In particular, we design and implement algorithms in order to check structural properties of graphs or generate them exhaustively and use computations to obtain examples and gain a better understanding of certain graph classes. These computational results can then in turn be used as a foundation for creating and proving mathematical theorems. We applied this methodology to several topics within graph theory, more specifically on topics in which *cycles* play a significant role. Cycles are an important concept within the theoretical study of graph theory, but also appear in a lot of applications such as the travelling salesperson problem, which is for example used in routing, logistics and genome assembly. Cycles also play a role in the design of fault-tolerant networks as they are a source of redundancy and many other such examples can be given. We divide the topics we study into three parts.

The first part is on K_2-*hypohamiltonian graphs*. These are graphs which are non-hamiltonian, but every subgraph obtained by deleting an edge and its two endpoints is hamiltonian. Their study is motivated by a long-standing and difficult conjecture of Grünbaum from 1974. We develop an algorithm for checking whether a graph satisfies this property and an algorithm which generates these graphs exhaustively. Using these results, we completely classify the orders for which K_2-hypohamiltonian graphs exist in the general case, as well as in the case of cubic graphs and snarks and describe or give lower bounds for smallest examples in the planar or cubic planar case.

The second part is on 2-*factors*, i.e. spanning subgraphs in which every vertex

has degree 2. A first work is on the *Frank number*, a parameter for 3-edge-connected graphs which refines k-edge-connectivity. An edge e of a graph G is *deletable* for some orientation if its restriction to $G-e$ is still strongly connected. The Frank number of G is defined as the minimum number of orientations of G such that every edge of G is deletable in at least one of the orientations. Hörsch and Szigeti showed that every 3-edge-connected graph G has Frank number at most 7 and if they are 3-edge-colourable that their Frank number is at most 3. We strengthen both of these results, by showing that every 3-edge-connected graph has Frank number at most 4 and 3-edge-connected 3-edge-colourable graphs have Frank number precisely 2. Moreover, we develop two sufficient conditions for a cyclically 4-edge-connected cubic graph to have Frank number 2, based on the structure of the 2-factors of the graph. Using these conditions we develop an algorithm for determining whether or not a graph has Frank number at most 2 and use it to determine that, apart from the Petersen graph, all cyclically 4-edge-connected cubic graphs up to order 36 have Frank number exactly 2.

A second work involving 2-factors is on *cycle permutation graphs*. These are cubic graphs of order n containing a 2-factor consisting of two induced cycles of length $n/2$. We develop a specialised algorithm for generating non-hamiltonian cycle permutation graphs and *permutation snarks*, i.e. non-3-edge-colourable cycle permutation graphs. On the one hand, we use the algorithm to completely settle a question posed by Klee in 1972 on the orders for which non-hamiltonian cycle permutation graphs exist. On the other hand, we improve on earlier computational results by Brinkmann et al. who generated permutation snarks up to order 34 using a generator-filter approach, by generating these graphs up to order 46. This gives computational evidence for the non-existence of permutation snarks on orders 6 mod 8 (which is still an open problem) and yields 736 extra examples in the class of cyclically 5-edge-connected permutation snarks, whereas only 13 such graphs had been generated in previous work.

The third part is on the *absence* of cycles. In a first work, we consider *HISTs*. These are spanning trees of a graph not containing any vertices of degree 2. We study these structures, which are antithetical to hamiltonian cycles, in the same spirit as hypohamiltonian graphs. We define *HIST-critical* graphs, which are graphs not containing a HIST, but every vertex-deleted subgraph does. We create an algorithm that checks whether a graph contains a HIST or whether it is HIST-critical and use this to give a near-characterisation of the orders for which HIST-critical graphs exist. We give an infinite family of planar HIST-critical graphs in which all but few vertices have degree 4, giving insight into the HIST-analogue of a theorem by Thomassen on planar hypohamiltonian graphs. We also provide computational evidence to a conjecture of Malkevitch stating that all 4-connected planar graphs contain a HIST.

A second work is on the fault-tolerance of networks. In certain applications the cost of a network is determined by their nodes of degree at least 3. Therefore spanning trees in which the number of branches and therefore the number of leaves is minimal are of interest. In this context we study what happens when a node fails. The *fault cost* is a measure of how many degree changes are required to obtain a new spanning tree with minimum number of leaves after a node is removed from the network. We develop an algorithm for determining the fault cost of a graph and use this to exhaustively determine the fault cost of small graphs. We show that any non-negative fault cost can occur and give infinite families of graphs with fault cost 1 and cubic graphs with fault cost 3.

These works show the merit of computational approaches in structural graph theory and that a good balance between computational and theoretical methods can be optimal for advancing the state-of-the-art research in an efficient way.

Beknopte samenvatting

Binnen toepassingen van grafentheorie zijn computationele methoden geen uitzondering, maar desondanks de groeiende populariteit van computermethoden in verscheidene domeinen en de forse toename van rekenkracht, worden zulke methoden vaak onderbenut binnen het domein van structurele grafentheorie.

In deze thesis proberen we de grenzen van state-of-the-art structureel grafentheorieonderzoek te verleggen door gebruik te maken van computationele methoden naast een meer theoretische aanpak. Meer bepaald ontwikkelen en implementeren we algoritmen om structurele eigenschappen van grafen na te gaan of om ze exhaustief te genereren. We gebruiken deze berekeningen om nieuwe voorbeelden te bekomen en meer inzicht te krijgen binnen bepaalde klassen van grafen. Deze computationele resultaten kunnen dan op hun beurt gebruikt worden als basis voor het ontwikkelen en bewijzen van wiskundige stellingen. We pasten deze methodologie toe op verschillende onderwerpen binnen de grafentheorie. Meer bepaald op onderwerpen waarin *cykels* een zekere rol spelen. Cykels zijn een belangrijk begrip binnen de theoretische studie van grafen, maar verschijnen ook in veel toepassingen zoals in het handelsreizigersprobleem, dat onder andere gebruikt wordt binnen routeplanning, logistiek en genoomassemblage. Cykels spelen ook een rol bij het ontwerpen van fouttolerante netwerken aangezien ze een bron van redundantie zijn. Verder zijn er nog veel andere dergelijke toepassingen van cykels. We verdelen de onderwerpen die we bestuderen op in drie categorieën.

De eerste categorie gaat over K_2-*hypohamiltoniaanse grafen*. Dit zijn grafen die niet-hamiltoniaans zijn, maar waarin elke deelgraaf bekomen door het verwijderen van een boog en zijn twee eindpunten wel hamiltoniaans is. De studie van deze grafen wordt gemotiveerd door een oud en moeilijk vermoeden van Grünbaum uit 1974. We ontwikkelen een algoritme om na te gaan of een graaf K_2-hypohamiltoniaans is en een algoritme dat deze grafen exhaustief genereert. Met behulp van de resultaten hieruit verkregen, classificeren we volledig de ordes waarvoor K_2-hamiltoniaanse grafen bestaan in het algemene

geval, alsook in het geval van kubische grafen en snarks. Verder beschrijven we of geven we ondergrenzen voor kleinste voorbeelden in het planaire of kubisch planaire geval.

De tweede categorie gaat over 2-*factoren*. Dit zijn opspannende deelgrafen waarin elke knoop graad 2 heeft. Een eerste werk in deze categorie gaat over het *Frank getal*, een parameter voor 3-boog-samenhangende grafen die de notie van k-boog-samenhang verfijnt. Een boog e in een graaf G is *verwijderbaar* voor een oriëntatie van G als de restrictie tot $G - e$ nog steeds sterk samenhangend is. Het Frank getal van G is gedefinieerd als het minimum aantal oriëntaties van G zodat elke boog in G verwijderbaar is in ten minste één van deze oriëntaties. Hörsch en Szigeti toonden aan dat elke 3-boog-samenhangende graaf Frank getal ten hoogste 7 heeft en wanneer ze 3-boog-kleurbaar zijn hun Frank getal ten hoogste 3 is. We versterken beide resultaten door aan te tonen dat elke 3-boog-samenhangende graaf Frank getal ten hoogste 4 heeft en 3-boog-samenhangende 3-boog-kleurbare grafen Frank getal precies 2 hebben. Bovendien ontwikkelen we twee voldoende voorwaarden voor een cyclisch 4-boog-samenhangende kubische graaf om Frank getal 2 te hebben, gebaseerd op de structuur van de 2-factoren van de graaf. Met deze voorwaarden ontwikkelen we een algoritme om te bepalen of een graaf Frank getal ten hoogste 2 heeft en gebruiken dit om aan te tonen dat, afgezien van de Petersen graaf, alle cyclisch 4-boog-samenhangende kubische grafen tot en met orde 36 Frank getal precies 2 hebben.

Een tweede werk over 2-factoren heeft te maken met *cykelpermutatiegrafen*. Dit zijn kubische grafen van orde n die een 2-factor bestaande uit twee geïnduceerde cykels van lengte $n/2$ bevatten. We ontwikkelen een gespecialiseerd algoritme voor het genereren van niet-hamiltoniaanse cykelpermutatiegrafen en permutatiesnarks, i.e. niet-3-boog-kleurbare cykelpermutatiegrafen. Enerzijds gebruiken we dit algoritme om een vraag gesteld door Klee in 1972 volledig te beantwoorden. Anderzijds verbeteren we eerdere computationele resultaten van Brinkmann et al. die permutatiesnarks tot en met orde 34 hebben gegenereerd met behulp van een generator-filteraanpak. Wij genereerden deze grafen tot en met orde 46. We bekrachtigen computationeel het ontbreken van permutatiesnarks op ordes $6 \bmod 8$ (dit is nog steeds een open probleem) en we produceren 736 extra voorbeelden in de klasse van cyclisch 5-boog-samenhangende grafen, terwijl hiervoor slechts 13 zulke voorbeelden gegenereerd waren in eerder werk.

De derde categorie betreft de *afwezigheid* van cykels. Een eerste werk binnen deze categorie gaat over *HISTs*. Dit zijn opspannende bomen in een graaf die geen knopen van graad 2 bevatten. We bestuderen deze structuren, antithetisch aan hamiltoniaanse cykels, op een manier gelijkaardig aan die voor hypohamiltoniaanse grafen. We definiëren *HIST-kritische* grafen, grafen die geen HIST bevatten, maar waarin elke deelgraaf bekomen door één knoop

te verwijderen wel een HIST bevat. We ontwikkelen een algoritme dat nagaat of een graaf een HIST bevat of HIST-kritisch is en gebruiken het om een bijna volledige karakterisering te geven van de ordes waarvoor HIST-kritische grafen bestaan. We geven ook een oneindige familie van HIST-kritische grafen waarin, behalve enkele, alle knopen graad 4 hebben en geven hierbij inzicht in een HIST-variant van een stelling door Thomassen over planaire hypohamiltoniaanse grafen. We bekrachtigen ook computationeel een vermoeden van Malkevitch dat zegt dat alle 4-samenhangende planaire grafen een HIST bevatten.

Een tweede werk gaat over de fouttolerantie van netwerken. In bepaalde toepassingen wordt de kost van een netwerk bepaald door de knopen van graad ten minste 3 (*vertakkingen*). Hierdoor zijn opspannende bomen waarin het aantal vertakkingen en dus ook het aantal bladeren minimaal zijn interessant. In deze context bestuderen we wat er gebeurt als er in een knoop een defect optreedt. De *foutkost* van een netwerk is een maat voor hoeveel graadwisselingen noodzakelijk zijn om een nieuwe opspannende boom met het minimum aantal bladeren te krijgen nadat een knoop verwijderd wordt uit het netwerk. We tonen aan dat elke niet-negatieve foutkost kan optreden en geven oneindige families van grafen met foutkost 1 en kubische grafen met foutkost 3.

Deze werken tonen de meerwaarde van een computationele aanpak binnen structurele grafentheorie aan en dat een goede balans tussen computationele en het theoretische methoden optimaal is om de grenzen van het state-of-the-art onderzoek te verleggen.

List of Symbols

$\chi'(G)$	chromatic index of G
$\Delta(G)$	maximum degree of G
$\delta(G)$	minimum degree of G
C_k	cycle of length k
$d_G(u)$	degree of vertex u in G
$E(G)$	edge set of G
$G[X]$	subgraph of G induced by $X \subseteq V(G)$
K_n	complete graph on n vertices
$N_G(u)$	neighbourhood of vertex u in G
$N_G[u]$	closed neighbourhood of vertex u in G
$V(G)$	vertex set of G

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graphs are mathematical structures encoding objects and whether these objects are in a pairwise relation with one another. They are defined by a set of vertices, representing the objects, and a set of edges, representing connections between pairs of vertices. Graphs can be used to model many things such as chemical molecules, road networks, communication networks, relationships within datasets, et cetera.

This thesis is situated in the domain of structural and computational graph theory, which focuses on investigating the structure of graphs both theoretically and algorithmically. We study these from a purely theoretical point of view, rather than focusing on the applications of graph theory and their details. A deeper understanding of the structural theory of graphs can lay the foundations for further theoretical study, but might also contribute to future research on applied graph theory. In this thesis we focus our scope to studying graphs in which the presence or absence of certain cycles plays an important role and develop and implement algorithms in order to help us answer problems in this domain. We focus on cycles as they can be considered to be one of the core topics of graph theory.

Informally, *cycles* are walks in a graph along the edges starting and ending at the same vertex and only passing through each vertex at most once. (For a precise definition see Section 1.1.) They are very important structures within graph theory and have been studied in widely different contexts. For one, Euler's paper [35] from 1736, generally considered to be the first paper on graph theory, already dealt with cycles. He studied a problem now famously known as the Seven Bridges of Königsberg, asking whether a walk through the city crossing each of the seven bridges exactly once was possible. One might ask the same

question for a walk starting and ending at the same point. Euler modelled the city as a graph and found a necessary condition for the former problem depending only on the number of edges (bridges) incident with each vertex (island). For the latter problem this same method is sufficient to determine whether or not such a walk, known as an *Eulerian cycle*, exists.

Another example is the study of *hamiltonian cycles*, which are cycles containing all vertices of the graph. (See Figure 1.1 for an example.) Checking whether a graph contains a hamiltonian cycle is computationally a lot more difficult than checking whether an Eulerian cycle exists. Indeed, in *connected* graphs, informally, graphs in which any vertex can be reached from any other, one can determine whether an Eulerian cycle exists by using a number of operations linear in the number of vertices in the graph. Determining whether or not a graph has a hamiltonian cycle is an example of an NP-complete problem [87]. For this class of problems no efficient (polynomial-time) algorithms with respect to the number of vertices are known and it is widely believed they do not exist.

Hamiltonian cycles are closely related to an important problem in theoretical computer science and operations research, known as the *travelling salesperson problem* or *TSP*. It asks, given a list of cities, to find a shortest route visiting all of them exactly once and ending at the starting point. In other words, to find a specific hamiltonian cycle in the graph modelling these cities. TSP has many direct applications such as in route planning, logistics and in the manufacturing of microchips. However, TSP also has less apparent applications such as in genome assembly, where DNA can be reconstructed from short sequencing reads. These reads are modelled by vertices and edges between vertices get a weight based on how much these reads overlap. The reconstruction can then be obtained by solving a TSP instance where you want to maximise the weights of your hamiltonian path. This famous approach, known as *Overlap-Layout-Consensus (OLC)*, was first introduced by Staden [124] in 1979.

On the theoretical side, there exists a lot of research studying classes of graphs guaranteed to contain hamiltonian cycles, examples include graphs satisfying certain conditions based on the degrees of its vertices [14, 34, 104], 4-connected graphs which are planar [133] or which avoid certain substructures [108]. See the survey [67] for more examples. On the other hand classes of graphs not containing hamiltonian cycles are also often studied, some examples include maximally non-hamiltonian graphs [13] and hypohamiltonian graphs [78]. We defer the definitions of these concepts to their respective papers.

More generally, however, various graph classes are defined by the presence or absence of cycles, for example *hamiltonian graphs* are graphs containing a hamiltonian cycle, *trees* are graphs containing no cycles, *bipartite graphs* are graphs containing no cycles of odd length, etc. Research related to cycles in

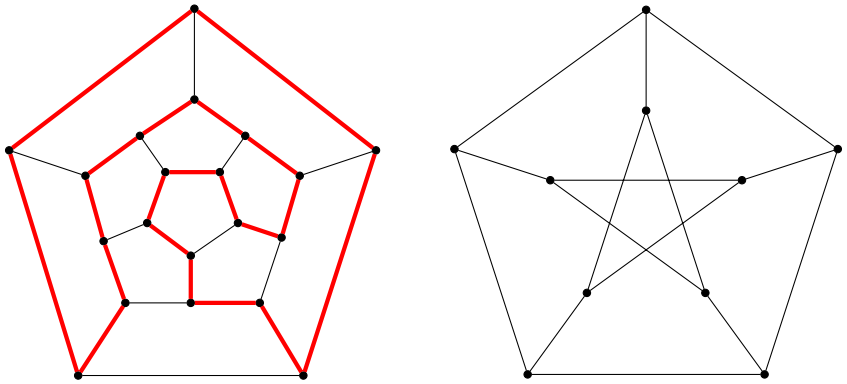


Figure 1.1: Left-hand side (a): The dodecahedron graph with a hamiltonian cycle marked in red. Right-hand side (b): The Petersen graph which has no hamiltonian cycle.

graph theory comes in many flavours. In this thesis we focus on the following three topics related to cycles.

The first is related to hamiltonian cycles and hypohamiltonian graphs. We study K_2-*hamiltonian graphs*, which are graphs for which every subgraph obtained by removing an edge and its endpoints is hamiltonian. If such a graph is non-hamiltonian, then it is a K_2-*hypohamiltonian graph*. The second topic can be seen as a generalisation of hamiltonian cycles. It concerns structures, known as 2-*factors*, consisting of (multiple) cycles that contain all vertices of the graph. A hamiltonian cycle is a 2-factor consisting of only one cycle. The third topic will look at structures that do not contain any cycles, i.e. *trees*, but which do contain all vertices, known as *spanning trees*. A more detailed overview together with some motivation of these topics will be given in Section 1.3.

In applications of graph theory it is quite common to use computational methods to deal with graphs. For example, in network analysis one may be interested in extracting features of the network, which due to their (typically) large size happens computationally. Algorithms that extract proximity information or capture local structure of the network can then be used in node classification. For example to detect bots in a social network [31]. In the timetabling problem one assigns certain events to certain timeslots subject to constraints. Typically this can be modelled as a graph and a feasible timetable corresponds to a feasible colouring of this graph. Solutions to graph colouring therefore imply solutions to problems which can be formulated in this way. Examples include, the rostering of nurses, scheduling of exams, assignment of tasks in a CPU, etc. In these applications one is typically more interested in obtaining a feasible

timetable, than in the properties of this graph. (Although knowledge of the structure of these graphs might yield faster algorithms.) This holds true in many of the applications. Graphs are a way to model the problem in order to be able to apply some algorithm to it in order to obtain some solution. However, the theoretical study of graph theory itself is still often done without the use of computational methods, but by strictly using theoretical techniques.

However, the use of computational methods can help in several ways. By developing and implementing algorithms that test whether a graph satisfies a certain property or that generate all graphs of a specific graph class, one might relatively quickly disprove a conjecture. These algorithms might also yield exhaustive lists of graphs in which relevant examples can be searched or give you more intuition about a given problem or a certain graph class. Computational methods might also help to check all cases of a proof which would be infeasible to check by hand.

A very famous example of the latter is the proof of the 4-colour theorem. Informally, the theorem states that any geographical map can be coloured with at most four colours in such a way that neighbouring regions do not share a colour. The conjecture originates from 1852, but was famously proven by Appel and Haken [4, 5] in the 1970's. Their proof reduced the problem to checking thousands of configurations which was done via computer. However, their result was quite controversial and contested by many mathematicians. In 1981, Schmidt published in his Master's thesis [118] that their proof contained a significant error. As a response to the claims that their proof contained flaws, Appel and Haken published their magnum opus [6]. A book containing a complete and detailed proof of the theorem, addressing and correcting also the error highlighted by Schmidt. This book included a 400 page microfiche supplement aiding the reader in checking all necessary cases. In more recent years, simplifications of the proof have arisen. One notable example is by Robertson, Sanders, Seymour and Thomas [113]. They not only reduce the number of cases that need to be checked by computer, but also algorithmically improve upon the quartic-time algorithm by Appel and Haken, by developing an algorithm that 4-colours planar maps in quadratic-time.

Another good example of computational methods in graph theory is the study of *snarks*, a specific class of graphs. (See Section 1.1 for their definition.) For many conjectures it holds that if they are true for snarks, they are true in general. In particular, for many of them it has even been proven that if a counterexample to the conjecture were to exist, a smallest one is a snark. Notable examples where this is the case include the Berge-Fulkerson conjecture [39], the Cycle Double Cover conjecture [121, 125] and Tutte's 5-flow conjecture [132], to which a lot of research has been devoted. However, up to 1975 only five non-trivial snarks were known [81]. Nowadays, algorithms have been developed to exhaustively

generate snarks and exhaustive generation of these graphs with a computer has been performed. This was done most recently by Brinkmann, Goedgebeur, Hägglund and Markström [18] in 2013, where they were able to generate all snarks up to 36 vertices (there are 432 105 682 such graphs). In their paper they disprove several conjectures by providing a counterexample generated by their algorithm.

While these are only two examples, they show that the use of computational methods can successfully prove or disprove state-of-the-art graph theoretical problems. Moreover, in some cases it is still impossible to prove the results which were obtained by computer by hand via analytical techniques as is the case for the 4-colour theorem. This indicates that a combination of both computational and analytical techniques are desirable when it comes to efficiently pushing forward the state-of-the-art in structural graph theory and is the motivation for the main research question of this thesis.

RQ: To what extent can computational methods contribute to extending the state-of-the-art research in structural graph theory on cycles?

While "structural graph theory on cycles" is quite a broad term, we believe our methods can be applied and contribute to the state-of-the-art research in many different topics within structural graph theory (not only on cycles). However, to restrict the scope slightly, all chapters present in this thesis will belong to this category of topics. In each of these chapters we only implicitly try to answer this general research question, but explicitly try to answer the open questions within each topic. We do this by attempting to push our computational methods as far as possible and seeing which advances this allows us to make to the respective state-of-the-art research.

While the introduction and conclusion of this thesis focuses on the computational aspects of the research, we want to highlight that our attempts to advance the state-of-the-art employs an interplay between both the computational methods as well as the theoretical methods. Using our developed algorithms, we can discover structural properties of the graphs under consideration. We then prove these properties theoretically and in turn we might use them to speed up the algorithm further, allowing us to discover more properties computationally, etc.

We restrict the thesis to three parts in which computational techniques had not yet been applied often and where we saw a lot of potential for this approach. In each of these topics, we study a different aspect of cycles in graph theory using computational means.

Before explaining these in more detail and in order to make this thesis more self-contained, we introduce some common definitions and preliminaries, most

of which are tacitly assumed to be known in each of the chapters. We follow this by giving a high-level description of some common ways to apply computational methods to structural graph theory in general and end the chapter with a more detailed overview of the content of this thesis.

1.1 Definitions and preliminaries

We now discuss some general notions on graph theory. The definitions and notations are for the most part standard throughout the field. For a textbook introducing graph theory as a whole, we refer to [33].

In this work we consider a *graph*, unless specified otherwise, to be a pair of sets $G = (V, E)$, such that $E \subseteq [V]^2$, i.e. E consists of 2-element subsets of G. These graphs are sometimes referred to as loopless, simple, undirected graphs. For such a graph G, we denote its set of *vertices* by $V(G)$ and its set of *edges* by $E(G)$. In this manuscript we assume all such sets are finite. For a set U, we denote its number of elements by $|U|$. The *order* of a graph G is $|V(G)|$ and its *size* is $|E(G)|$. We say G is a graph on $|V(G)|$ vertices.

Two vertices $u, v \in V(G)$ are *adjacent* if $\{u, v\} \in E(G)$. We say u and v are neighbours. We often write uv or (u, v) instead of $\{u, v\}$. A vertex u is *incident* with an edge e if $u \in e$. We also say e is incident with u in this case. For such an edge e, u is an *endpoint*. Two edges e and f are *adjacent* if they share an endpoint.

The *degree* of a vertex u is the number of edges incident with it. We denote it by $d_G(u)$ or simply $d(u)$ if it is clear from context in which graph we are considering this vertex. The *minimum* and *maximum degree* of a graph G are denoted by $\delta(G)$ and $\Delta(G)$, respectively. A graph in which the minimum and maximum degree coincide is called *regular*. If $r := \delta(G) = \Delta(G)$, we say G is r-regular. In particular, when $r = 3$ we say the graph is *cubic* and when $r = 4$, we say the graph is *quartic*. A graph in which every vertex has an even degree is called an *even graph*. The set of neighbours or *neighbourhood* of a vertex $u \in V(G)$ is denoted by $N_G(u)$ or $N(u)$. We write $N_G[U] := N_G(u) \cup \{u\}$ (or $N[u]$) for its *closed neighbourhood*.

We say a graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A subgraph H of G is *spanning* if $V(H) = V(G)$. We say a subgraph H is *induced* if for all $u, v \in V(H)$ if $uv \in E(G)$, then $uv \in E(H)$. For a subset $X \subseteq V(G)$, the subgraph of G *induced by* X, denoted $G[X]$, is the induced subgraph H of G with $V(H) = X$. If H is a subgraph of G, G is its *supergraph*.

A graph in which all vertices are pairwise adjacent is a *complete* graph. We denote a complete graph on n vertices by K_n. If G is a spanning subgraph of K_n and $E' \subseteq E(K_n)$, we write $G + E' := (V(G), V(E) \cup E')$. If $E' = \{e\}$, we might write $G + e$ instead of $G + \{e\}$.

If $V' \subseteq V(G)$ for a graph G, we define $G - V' := G[V(G) \setminus V']$ and might write $G - v$ instead of $G - \{v\}$ for some $v \in V(G)$. Similarly, if $E' \subseteq E(G)$, we define $G - E' := (V(G), E(G) \setminus E')$ and might write $G - e$ instead of $G - \{e\}$ for an edge $e \in E(G)$. If $X \subseteq V(G) \cup E(G)$, we say $G - X := (G - X \cap E(G)) - X \cap V(G)$.

We call two graphs G and G' *isomorphic* if there exists a bijection $f : V(G) \to V(G')$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(G')$. We call f an *isomorphism*. When $G = G'$, an isomorphism $f : V(G) \to V(G')$ is called an *automorphism*.

A path P is a non-empty graph (V, E) with $V = \{x_0, \dots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\}$, where all x_i are distinct. The *ends* of the path are x_0 and x_k and we say that P is a path between x_0 and x_k. We might write that P is an x_0x_k-path or simply an x_0-path or an x_k-path if only one of its ends is relevant. The *length* of P is its number of edges, i.e. k. We write $P = x_0 \dots x_k$. If $k \geq 3$, the graph $C := P + x_kx_0$ is a *cycle*. Its *length* is its number of edges. A cycle of length k is a *k-cycle* and often denoted by C_k. We also write $C = x_0 \dots x_k$ when it is clear C is a cycle.

If G has a cycle as a subgraph, we say its *girth* $g(G)$ is the length of a smallest such cycle. The maximum length of such a cycle is its *circumference*. An edge connecting two vertices of a cycle in G, but not belonging to the cycle is a *chord*. An *induced cycle* is a cycle which is an induced subgraph. It has no chords.

A graph G of order n is *hamiltonian* if it has a *hamiltonian cycle*, i.e. a cycle of length n. Then the circumference of G is n.

A graph G is *connected* if for any $u, v \in V(G)$ there exists a uv-path. A graph which is not connected is called *disconnected*. A maximal connected subgraph of G is a *component*. If $X \subseteq V(G) \cup E(G)$, then X *separates* G if $G - X$ is disconnected. If such a set X is minimal and $X \subseteq V(G)$, we call X a *(vertex-)cut of* G and if it is minimal and $X \subseteq E(G)$, we call X an *edge-cut* of G. A graph G of order n is *k-connected* for an integer $k > 0$ if $n > k$ and $G - V'$ is connected for every $V' \subseteq V(G)$ with $|V'| < k$. If G is k-connected but not $(k+1)$-connected we say G has *connectivity* k.

If G is a graph of order $n > 1$ and $G - E'$ is connected for every $E' \subseteq E(G)$ with $|E'| < k$, then G is *k-edge-connected*. If G is k-edge-connected but not $(k+1)$-edge-connected, it has *edge-connectivity* k.

If G is a graph of order $n > 1$ and $G - E'$ does not consist of multiple

components each containing a cycle for every $E' \subseteq E(G)$ with $|E'| < k$, then G is *cyclically k-edge-connected*. If G is cyclically k-edge-connected but not cyclically $(k+1)$-edge-connected, it has *cyclic edge-connectivity k*.

A (possibly disconnected) graph without any cycles is called *acyclic* or a *forest*. A connected forest is a *tree*. The vertices of degree 1 in a tree are its *leaves* and the vertices of degree at least 3 are its *branches*.

Definitions on *plane* and *planar* graphs are relatively easy to understand, but not easy to rigorously define without introducing some separate machinery. We therefore give an intuitive idea of what these graphs are and refer for a rigorous definition to Diestel's book on graph theory [33]. A *plane* graph is a drawing of a graph which is drawn in the plane without any crossing edges. A *planar* graph is an abstract graph (as we defined before), which can be drawn in the plane. A plane graph divides the plane into regions, called *faces*. While a planar graph a priori can be drawn in many different ways, a famous theorem by Whitney [139] states that if a planar graph is 3-connected, its faces are uniquely determined. Therefore, one can also speak about faces in the context of *polyhedral* graphs, i.e. 3-connected planar graphs. If a cycle in a polyhedral graph corresponds to the boundary of a face in a plane graph, we call it a *facial* cycle.

A *k-edge-colouring* of a graph G is a map $c : E(G) \to \{1, \ldots, k\}$ such that $c(u) \neq c(v)$ for any adjacent vertices u and v. If a graph admits a k-edge-colouring, we say it is *k-edge-colourable*. The *chromatic index* of G, denoted by $\chi'(G)$, is the smallest k for which a k-edge-colouring exists. A famous theorem by Vizing [137] states that $\chi'(G) \in \{\Delta(G), \Delta(G)+1\}$. In particular, a cubic graph is either 3-edge-colourable or non-3-edge-colourable, but 4-edge-colourable. Cubic graphs which are 2-edge-connected, i.e. *bridgeless*, and non-3-edge-colourable are called *snarks*. While some authors require that snarks are also cyclically 4-edge-connected or have girth at least 5, we will call all non-3-edge-colourable bridgeless cubic graphs snarks unless specified otherwise.

1.2 Computational methods

There are various different ways in which computational methods can be applied to problems in graph theory. We give a high-level sketch of some of the more common ones.

A first topic is that of determining whether a given graph admits a certain property. We might ask if a given graph G has maximum degree 5 for example, or if G has a hamiltonian cycle. In general, checking the former is quite easy and

can be done using only a number of operations linear in the order of the graph. In general, checking for a hamiltonian cycle is known to be NP-complete [40], meaning it is computationally a lot more difficult. However, most problems we consider later will be of this type, therefore we are less concerned with a problem's theoretical complexity, but more with the development and implementation of algorithms which are fast in practice. As we typically run these algorithms on all small graphs of a given graph class, we want the algorithms to be fast for this use case. (Of course, this does not mean theoretical complexity should be neglected as an algorithm running in exponential time will generally also run slower than one which is polynomial, even for our use case.) An algorithm that checks for a given graph whether it satisfies a certain property is often called a *filter* algorithm.

A second topic is exhaustive graph generation. Here, one is concerned with exhaustively generating all non-isomorphic graphs of a given order in a certain graph class up to as high as possible. As seen before, this method can be quite successful in disproving conjectures, as one can test a certain property on the generated elements of this graph class and determine exhaustively which of these graphs satisfies the property. If the class under consideration contains few known examples, performing exhaustive generation might yield more examples, which in turn can give more intuition about members of the class. In addition, graph generation is also useful for problems of a finite nature. For example, in extremal graph theory one is often interested in how small or big a given parameter can be (e.g. order of the graph) for graphs satisfying certain constraints. An example of this is Ramsey theory, where it is asked how many vertices a complete graph should have such that in every partition of the edges one of the classes always contains a certain substructure. This is quite a popular line of research in which computer searches have played an active role, especially in the last few decades and it has applications in various fields of mathematics as well as in information theory and theoretical computer science. (See the dynamic surveys [111] and [114].) Another example of this is the cage problem, where one is interested in finding *cages*, i.e. a smallest graph of a given girth and regularity. Also in this field exhaustive generation is employed in order to find specific cages or improve their lower or upper bounds. (See the dynamic survey [36].) Cages have for example applications in chemistry as they can model isoprenoid structures and benzenoid polycyclic hydrocarbons [9].

A lot of notable work has been done in the field of exhaustive generation and we will use several of the generators resulting from these works later. An example of this is `geng` from McKay's `nauty` library [100]. This is an exhaustive generator for graphs in general, but which also allows to restrict the generation to connected graphs, graphs without triangles, 4-cycles, 5-cycles, graphs with bounded minimum and maximum degree and graphs with various

other properties. Another example is `plantri` written by Brinkmann and McKay [21, 22, 23]. It allows for the generation of planar graphs with a given connectivity or minimum degree, planar graphs with a maximum number of edges and various other classes related to planar graphs. Finally, we also mention `snarkhunter` by Brinkmann, Goedgebeur and McKay [17, 18, 19]. A state-of-the-art generator for generating cubic graphs or snarks with a certain connectivity and lower bound on the girth. Such generators are quite useful when combined with a filter algorithm for a certain relevant property as it allows to quickly find examples having that property in the graph class one is interested in.

A third topic is that of isomorphism rejection and it is quite closely related to the second topic as one needs some form of isomorphism rejection in order to only output non-isomorphic graphs. When generating graphs it is undesirable to output isomorphic copies as these will share the exact same properties. The field of graph isomorphism is concerned with determining whether or not two graphs are isomorphic, also known as the *graph isomorphism problem*. It is widely studied both theoretically and practically and many tools for determining whether two graphs are isomorphic exist. (See [16, 70, 100] for surveys.) One of the most famous practical tools is McKay's `nauty` [100]. This tool actually computes a *canonical form* of a given graph. This is a specific labelling of the graph such that two graphs have the same canonical form if and only if they are isomorphic. The isomorphism test then consists of checking whether the canonical forms of each graph are identical. One can define many canonical forms. As an example, given an order on the vertices of the graph one can obtain an adjacency matrix, where each entry indicates whether or not two vertices are adjacent. Concatenating all rows and interpreting the result as a number, we get a total order on the labellings of the graph. We might define the canonical one to be the largest (or smallest) labelling of the graph. While we are not concerned with the actual graph isomorphism problem, we will often use McKay's program as a black box for rejecting isomorphic graphs in the remainder.

The theoretical side of the graph isomorphism problem is more concerned with its complexity and is one of the oldest and best studied algorithmic problems. It is currently unknown whether or not the graph isomorphism problem is solvable in polynomial time or if it is NP-complete. In 2017, Babai announced and quickly fixed an error in his earlier proof [8] and showed that one can solve this problem in quasi-polynomial time, i.e. there exists a constant c such that the algorithm on input size n runs in the worst case with an upper bound of $2^{O((\log n)^c)}$. His algorithm was a major breakthrough in the field.

Many other topics involving computational methods also exist such as extremal graph theory, automated conjecture generation, computational complexity

theory, etc. However, these are more out of the scope of this thesis.

A major concern when it comes to developing and implementing algorithms in a mathematical setting is that of correctness. How can one be sure that the computational results are correct? On the one hand, one can prove mathematically that a described algorithm is correct (which we indeed do). However, from an implementation perspective it is infeasible to prove this. To make things worse these implementations can often be quite extensive, for example our generator for cycle permutation graphs [64] consists of approximately 4 000 lines of C code. We mitigate the risk of implementation errors by performing extensive correctness tests. This comes in various forms. We typically verify as much as possible that our algorithm correctly computes known mathematical results. We also often use independent implementations and verify as much as feasible that both algorithms produce the same results. In all cases we make the implementation of our algorithms open source so that they can be inspected and verified by other researchers. While not a guarantee for correctness of the computation, as this is impossible to give, these checks do give strong evidence for it.

1.3 Overview of the thesis

We now describe each of the topics that will be considered during this thesis in more detail as well as the computational approaches we took in order to advance the respective state-of-the-art.

Part I of this thesis, consisting of Chapters 2 and 3, deals with hamiltonian cycles. Such cycles span all vertices of the given graph. In particular, we study K_2-*hamiltonian graphs*, i.e. graphs for which every subgraph obtained by deleting an edge together with its endpoints is hamiltonian. If the graphs themselves are non-hamiltonian, we call them K_2-*hypohamiltonian graphs*. These graphs are interesting due to their connection with a conjecture of Grünbaum [73], claiming that no graphs of order n exist with a circumference of $n - 2$, but in which any subgraph obtained by deleting a pair of vertices is hamiltonian. Moreover, these graphs are closely related to *hypohamiltonian* graphs, i.e. non-hamiltonian graphs in which every vertex-deleted subgraph is hamiltonian, which have been subject of study for a long time. These graphs were first introduced by Sousselier [123] in 1963 after which various papers by different authors followed, showing new infinite families of hypohamiltonian graphs or that they exist for certain orders. (See [78] for a survey.) In 1980, Grötschel [71] showed that certain hypohamiltonian graphs appear as facets of the travelling salesperson polytope. The complexity of deciding whether a graph is hypohamiltonian is

unknown (as is the case for K_2-hypohamiltonian graphs). However, Grötschel notes that the problem is most likely hard.

We show that K_2-hypohamiltonian graphs can behave quite differently than hypohamiltonian ones. A first algorithm we develop to this end, in Chapter 2, is a filter algorithm. This takes an input graph and determines whether or not it is K_2-hamiltonian or K_2-hypohamiltonian. We can produce complete lists of these graphs by combining our algorithm with a generator. For example, if we are interested in obtaining complete lists of K_2-hamiltonian connected graphs, we might use state-of-the-art generator `geng`, which is part of McKay's `nauty` library [100], to generate all pairwise non-isomorphic connected graphs of a given order and then apply our algorithm to determine which of these are K_2-hamiltonian. Similarly, if we are interested in K_2-hamiltonian snarks, we might use state-of-the-art program `snarkhunter` by Goedgebeur, Brinkmann, Hägglund and Markström [18] for the generation instead. However, if only a select number of graphs contains the property under consideration it is often a better idea to create a specialised generator for this class of graphs. We did this in Chapter 3, where we were able to obtain complete lists of K_2-hypohamiltonian graphs up to much higher orders. This allowed us to completely characterise all orders for which K_2-hypohamiltonian graphs exist as well as provide a new infinite family of K_2-hypohamiltonian graphs with vertices of high degree. This provides a first asymptotic result towards the question of how high the maximum degree of a K_2-hypohamiltonian graph can be.

Part II of this thesis, consisting of Chapters 4 and 5, deals with graphs in which multiple cycles span all vertices. In particular, in Chapter 4, we study the *Frank number*. This is a notion which was recently introduced by Hörsch and Szigeti [79] of which our study was the first to explore it computationally. An *orientation* of a graph is an assignment of exactly one direction to each edge of the graph. A theorem by Nash-Williams [101] states that $2k$-edge-connected graphs have an orientation which is k-arc-connected and vice versa. (Arc-connectivity is an analogue of edge-connectivity in directed graphs. See Chapter 4 for more details.) In order to say something about the connectivity of orientations in the case a graph is 3-edge-connected, one needs a finer notion. The *Frank number* of a graph G is the minimum number of orientations such that every edge of G can be removed in at least one of the orientations, while still preserving connectivity of this orientation in a directed sense. While the complexity of determining the Frank number of a graph G is unknown, determining whether for a given set of edges S there exists an orientation such that each $e \in S$ can be removed from G while preserving the connectivity of the orientation is NP-complete [79]. This gives some indication that determining the Frank number, or determining if it is equal to a given value k, seems to be computationally difficult. An exhaustive approach would need to check for the

input graph many of its orientations and compare every set of k such orientations with each other for the desired property. This process also in practice will quickly become infeasible, even for relatively small orders and small k. While this problem seemingly has nothing to do with cycles, we describe a sufficient condition when the given graph is built up of cycles in a specific way. Checking this condition first allows us to speed up the computations significantly.

In Chapter 5, we are interested in generating *cycle permutation graphs*. These are cubic graphs consisting of two cycles of the same length without chords. In particular, cycle permutation graphs which are snarks, i.e. *permutation snarks* are interesting due to their close connection with various difficult conjectures such as Tutte's 5-flow conjecture [132], the Cycle Double Cover conjecture [121, 125], the Berge-Fulkerson conjecture [39] and the Alon-Tarsi 7/5-conjecture [3]. For each of these conjectures a smallest counterexample, if it exists, is a snark. Non-hamiltonian cycle permutation graphs and permutation snarks have been studied computationally before, but not yet using a specialised approach. One of these computational works is by the famous mathematician Victor Klee, who was active in various fields, such as in convexity, functional analysis, analysis of algorithms and combinatorics. In 1972, Klee [89] was interested in the hamiltonicity of cycle permutation graphs as they are a class of graphs always containing at least $2n$ hamiltonian paths, where n is the order of the graph, but which are not necessarily hamiltonian. He asked for which orders non-hamiltonian cycle permutation graphs exist, but only managed to prove this for orders 0 mod 4. Klee also mentions an alternative way of proving some of his results, by making use of the fact that a certain graph class (a subset of the generalised Petersen graphs) is known to be hypohamiltonian.

In this work, we create a specialised generator to obtain all pairwise non-isomorphic cycle permutation graphs as well as non-hamiltonian cycle permutation graphs and cycle permutation graphs which are snarks, i.e. *permutation snarks*. Using the results from this algorithm we completely settle the question posed by Klee [89] in 1972. Moreover, we generate permutation snarks up to order 46, improving earlier computational results by Brinkmann, Goedgebeur, Hägglund and Markström [18], who generated them up to order 34 using a generator-filter approach. We hereby add 736 extra graphs to the 13 previously generated examples in the important class of cyclically 5-edge-connected permutation snarks. It was previously conjectured by Zhang [148] that the Petersen graph is the only graph in this class. We also provide computational evidence for the non-existence of permutation snarks on orders 6 mod 8, one of the main open problems concerning permutation snarks.

Part III of this thesis, consisting of Chapters 6 and 7, deals with structures that do *not* contain any cycles. In Chapter 6, we study spanning trees which do not contain any vertices of degree 2. Such graphs are sometimes called

homeomorphically irreducible and such spanning trees are typically abbreviated to *HISTs*. One can consider them to be the opposite of hamiltonian cycles (connected spanning subgraphs in which every vertex has degree 2). Our study continues in the same vein as K_2-hamiltonian graphs (cf. Part I), by asking for the existence of HISTs in large (vertex-deleted) subgraphs rather than the existence of hamiltonian cycles in large (K_2-deleted) subgraphs. Despite the fact that the enumeration of homeomorphically irreducible trees had already been studied in 1959 [74], we believe we are the first to do so computationally. To this end we wrote a filter program that determines if a graph is a so-called *HIST-critical graph*, i.e. does not contain a HIST, but every vertex-deleted subgraph does.

Using the results obtained by our generator, we give a near-complete characterisation of orders for which HIST-critical graphs exist. We also give an infinite family of planar HIST-critical graphs in which nearly all vertices have degree 4 in an attempt to construct 4-regular examples. (Thomassen has shown that planar hypohamiltonian graphs must always contain a cubic vertex [130].) Finally, we use our algorithm to computationally verify a conjecture by Malkevitch [97]. He was interested in the degree sequences of spanning trees of planar graphs and shows that in certain cases vertices of degree 2 must always be present in such spanning trees. His conjecture states, however, that 4-connected planar graphs always contain a HIST. We give a result on a family of 4-connected planar graphs containing few HISTs.

Chapter 7 concerns a theoretical study of another concept related to spanning trees, which is motivated by applications. In certain optimisation problems concerning spanning trees, one is interested, in order to minimise costs, to find those with few leaves (which also implies a bound on the number of branches). For example, in [41] the authors study a problem in optical network design, where a light-tree connects one node in the network to a set of other nodes allowing multicast communication. The branches in this network correspond to switches that are able to split light and the number of these more sophisticated and expensive switches is limited. The number of leaves in a spanning tree with as few leaves as possible is called the *minimum leaf number*. We study such networks with respect to their fault-tolerance. More precisely, we investigate what happens when a certain node in the network has a fault, i.e. is removed from the network. On the one hand, we study graphs in which such a fault does not increase the number of leaves. These are the so called *leaf-guaranteed* graphs. On the other hand, we consider the *fault cost* of a graph. This is a general tool that tries to gauge how well such a network with deals with degree changes in its minimum spanning trees due to the loss of a node.

We develop an algorithm for determining the fault cost of a graph and perform an exhaustive exploration of the fault cost of small graphs. We also derive several

infinite families showing that any non-negative integer can occur as a fault cost and provide infinitely many graphs with fault cost 1, despite these seemingly being hard to find. We also show some basic properties of leaf-guaranteed graphs and show that any network can be contained in a leaf-guaranteed one.

In the following six chapters, we report how we extended the state-of-the-art research in each of these topics. Each chapter is nearly identical to works published or submitted by myself and co-authors. Most changes made to these works are to create a cohesive style in this manuscript.

In Chapter 8, we reflect on the obtained results, how they contribute to answering the central research question and give directions for future work.

Part I

K_2-Hamiltonian Graphs

Chapter 2

K_2-Hamiltonian Graphs

This chapter is a minimally edited version of the following published work.

J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. K_2-hamiltonian graphs: II. *J. Graph Theory*, 105(4):580–611, 2024. DOI: 10.1002/jgt.23057

My contributions to this paper, in accordance with the contributor role taxonomy (CRediT)[1], consist of: Conceptualisation, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review and Editing, and Visualisation.

2.1 Introduction

We shall call a graph K_1-*hamiltonian* if all of its vertex-deleted subgraphs are hamiltonian and K_2-*hamiltonian* if the removal of any pair of adjacent vertices yields a hamiltonian graph. Non-hamiltonian K_1-hamiltonian graphs are called *hypohamiltonian* and, in the same vein, we say that a graph is K_2-*hypohamiltonian* if it is K_2-hamiltonian yet non-hamiltonian. Here we will mainly focus on properties of K_2-hypohamiltonian graphs, but we will also discuss their interplay with hamiltonicity and K_1-hamiltonicity. This paper is a continuation of [143]; for a motivation of our interest in K_2-hamiltonian graphs we refer to that article's introduction, and references therein.

An n-vertex K_2-hamiltonian graph may contain a hamiltonian cycle but no $(n-1)$-cycle (e.g. a d-dimensional cube for $d \geq 3$), but it may also contain

[1] See https://credit.niso.org/.

no hamiltonian cycle yet be K_1-hamiltonian, e.g. the Petersen graph. As we shall see, the Petersen graph is in fact the smallest K_2-hypohamiltonian graph. Coxeter's graph, another famous hypohamiltonian graph, is not K_2-hamiltonian. Furthermore, every planar 4-connected graph is hamiltonian, K_1-hamiltonian, K_2-hamiltonian, and indeed even K_3-*hamiltonian* (i.e. the removal of any triangle yields a hamiltonian graph) by work of Tutte [133], Nelson [102], Thomas and Yu [126], and Sanders [117]. The reader might wonder if a K_2-hamiltonian graph on n vertices and containing neither n- nor $(n-1)$-cycles exists; this is a good question to which we do not know the answer. It is a special but interesting case of a conjecture of Grünbaum [73] which states that for any integer $k \geq 2$, there exists no graph whose order and circumference differ by k and in which any set of k vertices is avoided by some longest cycle. A relaxation of this problem of Grünbaum is also still open: Katona, Kostochka, Pach, and Stechkin [88] asked whether an n-vertex graph in which any $n-2$ vertices induce a hamiltonian graph, must itself be hamiltonian. (In fact, Katona et al. had raised this problem far more generally; one obtains their original question by replacing "$n-2$" with k for $n/2 < k < n-1$, where n denotes the graph's order.) Van Aardt, Burger, Frick, Llano, and Zuazua raised in [136] an equivalent problem; see their Question 1.

It follows from a result of Thomason [127] that if a cubic graph contains a vertex-deleted subgraph which has an odd number of hamiltonian cycles, then the graph itself must be hamiltonian; and that if a cubic graph has an odd number of hamiltonian cycles, it is K_1-hamiltonian. The first statement is not true if we replace K_1 by K_2: simply take the Petersen graph. It is cubic and non-hamiltonian, but removing adjacent vertices yields a graph containing exactly one hamiltonian cycle. The second statement's K_2-variant is not true either: take any graph obtained from K_4 by subsequently replacing vertices by triangles. The resulting graph contains exactly three hamiltonian cycles, but also a triangle whose vertices are cubic, so it cannot be K_2-hamiltonian (see [143, Proposition 1]).

In Section 2.2 we address, as announced in [143], the natural question whether planar K_2-hypohamiltonian graphs exist. Grünbaum had conjectured [73] that every planar K_1-hamiltonian graph is hamiltonian, but Thomassen [130] disproved this conjecture. On the other hand, in a subsequent paper, Thomassen showed that every planar K_1-hamiltonian graph *without cubic vertices* is hamiltonian [128], thereby giving an elegant amendment of Grünbaum's conjecture so that it does hold. It is natural to wonder about other ways in which this conjecture can be "saved". We will prove that by replacing K_1 with K_2 one *cannot* save the conjecture, i.e. it is not true that every planar graph in which every K_2-deleted subgraph is hamiltonian, must itself be hamiltonian. Moreover, as we shall see, in a planar graph not even the hamiltonicity of

every K_1-deleted *and* K_2-deleted subgraph necessarily implies the hamiltonicity of the graph itself. In fact, we will even show that the number of planar hypohamiltonian K_2-hamiltonian graphs grows at least exponentially, giving an alternative proof of a result of Collier and Schmeichel [29].

In Section 2.3 we determine all K_2-hypohamiltonian graphs up to a given order for specific classes of graphs (i.e. general graphs, cubic graphs, snarks, planar graphs, cubic planar graphs) and characterise the orders for which they exist using a combination of computational methods and theoretical arguments.

Finally, in Section 2.4, we conclude the paper with an erratum concerning [143] and discuss potential future directions of research.

We end this section with the introduction of definitions and notations that will be used throughout the remainder of the article. We refer to [33] for any standard graph theory terminology which is not explicitly defined, but note that here we write K_n (not K^n) for the n-vertex complete graph.

For a set S, we say that $A, B \subset S$ *partition* S if $A \cap B = \emptyset$ and $A \cup B = S$. In a graph G, consider $S \subset V(G)$. Then the *induced subgraph* $G[S]$ is the graph whose vertex set is S and whose edge set consists of all of the edges in $E(G)$ with both endpoints in S. Let G be a non-complete graph of connectivity k and order greater than k, X a k-cut in G, and C a component of $G - X$. Then $G[V(C) \cup X]$ is an *X-fragment* of G with *attachments* X. An X-fragment is *trivial* if it contains exactly $|X|+1$ vertices. Sometimes we wish to emphasise the cardinality of the set X of attachments, so we also say that $G[V(C) \cup X]$ is an *$|X|$-fragment*. Let F, F' be disjoint 3-fragments of graphs of connectivity 3, and let F have attachments x_1, x_2, x_3 and F' have attachments x'_1, x'_2, x'_3. Identifying x_i with x'_i for all i, we obtain the graph $(F, \{x_1, x_2, x_3\})\,(F', \{x'_1, x'_2, x'_3\})$. If each of two disjoint fragments admits a planar embedding with cofacial attachments, it is easy to see that the identifications from the aforementioned operation can be applied such that the resulting graph is planar, as well. A cut X of G is *trivial* if $G - X$ has exactly two components, one of which is K_1. A path with end-vertex v is a *v-path*, and a v-path with end-vertex $w \neq v$ is a *vw-path*.

2.2 Variations of a conjecture of Grünbaum

Motivated by Grünbaum's conjecture that every planar K_1-hamiltonian graph is hamiltonian [73] and Thomassen's elegant way to save it [128], we now give two proofs of the fact that even if in a planar graph the deletion of any single vertex *and* the deletion of any pair of adjacent vertices yields a hamiltonian graph, the presence of a hamiltonian cycle is not guaranteed—a property shared

by the Petersen graph. We give infinitely many examples, and note that they in fact satisfy the stronger property that any complete graph we remove, we get a hamiltonian graph (simply because they contain no K_t for all integers $t \geq 3$). For our first proof we introduce a new method inspired by an old idea of Chvátal—his so-called *flip-flop* graphs [28]—while the second proof is based on a lemma from the first article in this series [143]. This second proof is shorter, but the first one builds a framework which we believe could be useful in attacking related problems concerning longest paths and longest cycles, in particular regarding graphs of minimum degree 4 and maybe even the notoriously difficult 4-connected case.

Let G be a graph and let $a, b, c, d \in V(G)$ be pairwise distinct. The pair of vertices (a, b) is said to be *good* if there is a hamiltonian ab-path in G, and the pair of pairs of vertices $((a, b), (c, d))$ is *good* if there exist an ab-path and a cd-path in G whose vertex sets partition $V(G)$. Otherwise, in either case, the pair is said to be *bad*. The quintuple (G, a, b, c, d) is a *cell* if G is a graph and $a, b, c, d \in V(G)$ are pairwise distinct. These four vertices shall be called *outer vertices* and every vertex that is not an outer vertex is an *inner vertex*. A cell is *suitable* if all of the following properties hold.

1.1. The pairs $(a, b), (a, c), (b, d)$, and (c, d) are good.

1.2. The pairs (a, d) and (b, c) are bad.

1.3. The pair of pairs $((a, b), (c, d))$ is good.

1.4. The pairs of pairs $((a, d), (b, c))$ and $((a, c), (b, d))$ are bad.

1.5(a). For any outer vertex v, the pairs $(a, b), (a, c), (b, d)$, and (c, d) which do not contain v as one of the vertices are bad in $G - v$.

1.5(b). For any outer vertex v, the pairs (a, d) and (b, c) which do not contain v as one of the vertices are bad in $G - v$.

1.6. Deleting any of the pairs of vertices (a, c), (a, d), (b, c), (b, d), the remaining pair of outer vertices is good in the resulting graph (e.g. the pair (b, d) is good in $G - a - c$, etc.).

Property 1.5 states that, for any outer vertex v, all pairs of outer vertices are bad in $G - v$. The distinction between cases (a) and (b) is needed in the following definitions. A suitable cell (G, a, b, c, d) is a K_1*-cell* if by deleting any inner vertex of G we obtain a graph in which at least one of the bad pairs stated in Properties 1.2, 1.4, 1.5(a) becomes good. A suitable cell (G, a, b, c, d) is a K_2*-cell* if all of the following properties hold.

2.1. By deleting any two neighbouring inner vertices of G we obtain a graph in which at least one of the bad pairs stated in Properties 1.2, 1.4, 1.5(a) becomes good.

2.2. If $x \in N(a)$, then (b, c) is good in $G - a - x$.
2.3. If $x \in N(d)$, then (b, c) is good in $G - d - x$.
2.4. If $x \in N(b)$, then (a, d) is good in $G - b - x$.
2.5. If $x \in N(c)$, then (a, d) is good in $G - c - x$.

It is far from obvious whether K_1- or K_2-cells actually exist; we shall later see examples of K_1- and K_2-cells, and even cells that are both K_1- and K_2-cells. On the other hand, it is easy to verify that the underlying graph of a suitable cell must have at least five vertices.

Let $k \geq 3$ be a fixed but arbitrary integer. For $i = 0, \ldots, k-1$, consider pairwise disjoint cells $H_i = (G_i, a_i, b_i, c_i, d_i)$. Henceforth, indices are taken mod k. For a fixed but arbitrary $j \in \{1, 2\}$, if H_i is a K_j-cell for every i, then the graph Γ_j^k is defined as $\bigcup_i G_i$ in which we identify b_i with a_{i+1} and c_i with d_{i+1}, for $i = 0, \ldots, k-1$. Note that if H_i admits a planar embedding in which a, b, c, d occur in the same facial cycle and in this order, for $i = 0, \ldots, k-1$, then the aforementioned identifications can be performed such that Γ_j^k is planar, as well. Put $V_i := V(G_i)$, $E_i := E(G_i)$, and $P_i := \{a_i, b_i, c_i, d_i\}$; vertices in P_i are *outer vertices* (of G_i), while vertices in $V_i \setminus P_i$ are *inner vertices* (of G_i).

Theorem 2.1. *For every odd integer $k \geq 3$ the graph Γ_1^k is hypohamiltonian and the graph Γ_2^k is K_2-hypohamiltonian.*

Proof. We will first show that both Γ_1^k and Γ_2^k are non-hamiltonian. Let us assume that Γ_1^k has a hamiltonian cycle $\mathfrak{h}$—the proof will be the same for Γ_2^k, since at this point we shall only use properties of suitable cells—and let us consider $\mathfrak{h}_i := \mathfrak{h}[V_i]$. All vertices of $\mathfrak{h}_i$ have degree at most 2 and the inner vertices (of G_i) must have degree 2 in $\mathfrak{h}_i$, while at least two of the outer vertices have degree 1. It is also straightforward to see that each component of $\mathfrak{h}_i$ contains an outer vertex. Therefore, exactly one of the following cases must hold.

Case 1. Two outer vertices have degree 1 and the other two outer vertices have degree 0 in $\mathfrak{h}_i$. Then $\mathfrak{h}_i$ consists of a path $\mathfrak{p}_i$ between two outer vertices and two isolated (outer) vertices.

Case 2. Two outer vertices have degree 1, the others have degree 0 and 2, respectively. Then $\mathfrak{h}_i$ consists of a path $\mathfrak{p}_i$ between two outer vertices and an isolated (outer) vertex.

Case 3. Two outer vertices have degree 1, the others have degree 2. Then $\mathfrak{h}_i$ is a path between two outer vertices.

Case 4. All four outer vertices have degree 1. Then $\mathfrak{h}_i$ is the disjoint union of two paths, both between two outer vertices.

We now treat these four cases. Case 2 is easy to handle: it cannot occur due to Property 1.5.

In Case 4, $\mathfrak{h}_i$ must be the union of an a_ib_i- and a c_id_i-path, because of Property 1.4. Now we show that Case 4 is also impossible. Assume to the contrary that Case 4 holds for some $\mathfrak{h}_i$. Then there exists an index j such that Case 4 holds for $\mathfrak{h}_j$, but not for $\mathfrak{h}_{j+1}$. Indeed, if such a j did not exist then $\mathfrak{h}$ would not be connected, but the union of two cycles, since every $\mathfrak{h}_i$ is the union of an a_ib_i- and a c_id_i-path. For $\mathfrak{h}_{j+1}$ either Case 1 or Case 3 must hold and the degree 1 vertices of $\mathfrak{h}_{j+1}$ are a_{j+1} and d_{j+1}. Case 3 now cannot hold, because then (a_{j+1}, d_{j+1}) would be a good pair in G_{j+1}, contradicting Property 1.2. Thus Case 1 holds, that is b_{j+1} and c_{j+1} are isolated vertices in $\mathfrak{h}_{j+1}$. Then both a_{j+2} and d_{j+2} have degree 2 in $\mathfrak{h}_{j+2}$, which is also impossible, since then $\mathfrak{h}_{j+2}$ would be a hamiltonian $b_{j+2}c_{j+2}$-path in G_{j+2}, contradicting Property 1.2 ($\mathfrak{h}_{j+2}$ exists, since $k \geq 3$).

Now we prove that if $\mathfrak{h}_i$ corresponds to Case 1 then $\mathfrak{h}_{i+1}$ corresponds to Case 3 and vice versa. Let us assume first that $\mathfrak{h}_i$ corresponds to Case 1. The end-vertices of $\mathfrak{p}_i$ cannot be a_i and d_i, as otherwise (b_{j+1}, c_{j+1}) would be a good pair in G_{j+1}, contradicting Property 1.2. Similarly, the end-vertices of $\mathfrak{p}_i$ cannot be b_i and c_i, either. Thus, either b_i or c_i must be an isolated vertex of $\mathfrak{h}_i$, so either $a_{i+1} = b_i$ or $d_{i+1} = c_i$ has degree 2 in $\mathfrak{h}_{i+1}$. Therefore $\mathfrak{h}_{i+1}$ cannot correspond to Case 1 and we have already seen that it cannot correspond to Case 2 or 4. Now let us assume that $\mathfrak{h}_i$ corresponds to Case 3. Then the end-vertices of $\mathfrak{h}_i$ cannot be a_i and d_i, and they cannot be b_i and c_i, by Property 1.2. Thus, either b_i or c_i has degree 2 in $\mathfrak{h}_i$, whence, either $a_{i+1} = b_i$ or $d_{i+1} = c_i$ are isolated vertices in $\mathfrak{h}_{i+1}$. Therefore $\mathfrak{h}_{i+1}$ cannot correspond to Case 3 and we have already seen that it cannot correspond to Case 2 or 4.

We have proven that Cases 1 and 3 alternate among the $\mathfrak{h}_i$'s, thus the number k of $\mathfrak{h}_i$'s should be even. Since this is not the case, we have finished the proof of the non-hamiltonicity of Γ_1^k (and also Γ_2^k).

We proceed by proving that Γ_1^k is hypohamiltonian. Since Γ_1^k is non-hamiltonian, it remains to show that for every vertex x, the graph $\Gamma_1^k - x$ is hamiltonian. Let x be an arbitrary vertex of Γ_1^k and let us assume without loss of generality that $x \in V_0$. Suppose first that x is an inner vertex of G_0. By the definition of K_1-cells, for $G_0 - x$ one of the pairs appearing in Properties 1.2, 1.4, 1.5(a) must be good.

Case 1. One of the pairs appearing in Property 1.2 becomes good after deleting

x from G_0. We may assume without loss of generality that (b_0, c_0) is good in $G_0 - x$, i.e. there exists a hamiltonian b_0c_0-path $\mathfrak{p}_0$ in $G_0 - x$. For $i = 1, \ldots, k-2$ let $\mathfrak{u}_i$ be the union of the a_ib_i- and c_id_i-paths in G_i guaranteed by Property 1.3 . Let furthermore $\mathfrak{p}_{k-1}$ be the hamiltonian $a_{k-1}d_{k-1}$-path of $G_{k-1} - b_{k-1} - c_{k-1}$ guaranteed by Property 1.6. Then $\mathfrak{p}_0 \cup \mathfrak{u}_1 \cup \ldots \cup \mathfrak{u}_{k-2} \cup \mathfrak{p}_{k-1}$ is a hamiltonian cycle of $\Gamma_1^k - x$.

Case 2. One of the pair of pairs $((a_0, d_0), (b_0, c_0))$ and $((a_0, c_0), (b_0, d_0))$ appearing in Property 1.4 becomes good after deleting x from G_0, i.e. either there exists an a_0d_0-path P and a b_0c_0-path Q which together partition $V_0 - x$ or there exists an a_0c_0-path P and a b_0d_0-path Q which together partition $V_0 - x$. For both cases let us define $\mathfrak{u}_i$ for $i = 1, \ldots, k-1$ as the union of the a_ib_i- and c_id_i-paths in G_i guaranteed by Property 1.3. Then $P \cup Q \cup \mathfrak{u}_1 \cup \ldots \cup \mathfrak{u}_{k-1}$ is a hamiltonian cycle of $\Gamma_1^k - x$.

Case 3. One of the pairs appearing in Property 1.5(a) becomes good after deleting x from G_0. We may assume without loss of generality that (a_0, b_0) or (a_0, c_0) is good in $G_0 - d_0 - x$.

Case 3a. There exists a hamiltonian a_0c_0-path $\mathfrak{p}_0$ in $G_0 - d_0 - x$. For all odd $j = 1, \ldots, k-2$, let $\mathfrak{p}_j$ be a hamiltonian b_jd_j-path of $G_j - a_j - c_j$, guaranteed by Property 1.6. If $k \geq 5$, for all even $j = 2, \ldots, k-3$ let $\mathfrak{p}_j$ be a hamiltonian a_jc_j-path of G_j, guaranteed by Property 1.1. Finally, let $\mathfrak{p}_{k-1}$ be a hamiltonian $a_{k-1}b_{k-1}$-path of G_{k-1}, again guaranteed by Property 1.1. Now $\mathfrak{p}_0 \cup \ldots \cup \mathfrak{p}_{k-1}$ is a hamiltonian cycle of $\Gamma_1^k - x$.

Case 3b. There exists a hamiltonian a_0b_0-path $\mathfrak{p}_0$ in $G_0 - d_0 - x$. For all odd $j = 1, \ldots, k-2$, let $\mathfrak{p}_j$ be a hamiltonian a_jc_j-path of $G_j - b_j - d_j$, guaranteed by Property 1.6. For all even $j = 2, \ldots, k-1$, let $\mathfrak{p}_j$ be a hamiltonian b_jd_j-path of G_j, guaranteed by Property 1.1. Now $\mathfrak{p}_0 \cup \ldots \cup \mathfrak{p}_{k-1}$ is a hamiltonian cycle of $\Gamma_1^k - x$.

Let us suppose now that x is an outer vertex of G_0; without loss of generality assume $x = a_0$. There is a hamiltonian b_0d_0-path $\mathfrak{p}_0$ in $G_0 - a_0 - c_0$ by Property 1.6. Let $\mathfrak{p}_1$ be a hamiltonian a_1b_1-path of G_1 (which exists by Property 1.1) and for all even $j = 2, \ldots, k-1$ let $\mathfrak{p}_j$ be a hamiltonian a_jc_j-path of $G_j - b_j - d_j$, guaranteed again by Property 1.6. If $k \geq 5$, for all odd $j = 3, \ldots, k-2$, let furthermore $\mathfrak{p}_j$ be a hamiltonian b_jd_j-path of G_j, guaranteed again by Property 1.1. Then $\mathfrak{p}_0 \cup \ldots \cup \mathfrak{p}_{k-1}$ is a hamiltonian cycle of $\Gamma_1^k - a_0$, finishing the proof of the hypohamiltonicity of Γ_1^k.

Since Γ_2^k is non-hamiltonian, we still have to show that for any neighbouring vertices x and y, the graph $\Gamma_2^k - x - y$ is hamiltonian. By the construction of

Γ_2^k there exists a G_i which contains both x and y, so we may assume without loss of generality that $x, y \in V_0$.

If both x and y are inner vertices of G_0, then by Property 2.1, one of the pairs appearing in Properties 1.2, 1.4, 1.5(a) is good in $G_0 - x - y$ and the proof of the hamiltonicity of $\Gamma_2^k - x - y$ is the same as the proof of the hamiltonicity of $\Gamma_1^k - x$ (where x is an inner vertex of G_0) we have just seen.

Let us suppose now that y is an outer vertex of G_0 and $x \in V_0$ is an arbitrary, but fixed neighbour of y. Without loss of generality we may assume $y = a_0$. By Property 2.2 there exists a hamiltonian b_0c_0-path $\mathfrak{p}_0$ in $G_0 - a_0 - x$. Now the proof is essentially the same as for Case 1 of the proof of the hypohamiltonicity of Γ_1^k, but we include it here, since the setting is a slightly different one.

For $i = 1, \ldots, k-2$ let $\mathfrak{u}_i$ be the union of the a_ib_i- and c_id_i-paths of G_i guaranteed by Property 1.3. Let furthermore $\mathfrak{p}_{k-1}$ be a hamiltonian $a_{k-1}d_{k-1}$-path of $G_{k-1} - b_{k-1} - c_{k-1}$ whose existence is guaranteed by Property 1.6. Now $\mathfrak{p}_0 \cup \mathfrak{u}_1 \cup \ldots \cup \mathfrak{u}_{k-2} \cup \mathfrak{p}_{k-1}$ is a hamiltonian cycle of $\Gamma_2^k - a_0 - x$, finishing the proof of the K_2-hypohamiltonicity of Γ_1^k.

□

We now present a suitable cell that is both a K_1- and K_2-cell: the quintuple (J_{18}, a, b, c, d) of Figure 2.1. It is obtained by removing two adjacent vertices from the dodecahedron. This particular graph was already used by Faulkner and Younger in the context of determining hamiltonian properties of planar 3-connected cubic graphs [38]. Although they mention two of the properties used below (more precisely, points 1 and 5), no proof is given; we therefore give all details here.

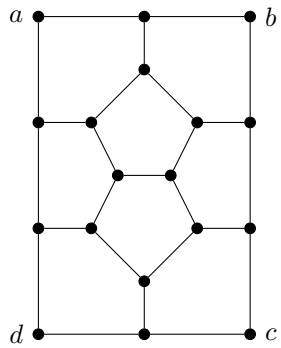


Figure 2.1: The graph J_{18}.

Lemma 2.2. *The quintuple* (J_{18}, a, b, c, d) *of Figure 2.1 is a suitable cell.*

Proof. We have to verify Properties 1.1–1.6. Some of the upcoming arguments are most easily verified by using figures, which we have provided in the Appendix; all figures referenced in this proof are to be found there. Ignoring symmetric cases, for Property 1.1 it suffices to provide a hamiltonian ab-path and a hamiltonian ac-path in J_{18}, see Figure A.1 in the Appendix. For Property 1.3 it is enough to present an ab-path and a cd-path whose vertex sets partition $V(J_{18})$; an easy task, verified in Figure A.1. Again by symmetry, for Property 1.6 it suffices to give a hamiltonian bc-path in $J_{18} - a - d$ and a hamiltonian bd-path in $J_{18} - a - c$; these are depicted in Figure A.2 in the Appendix. For Properties 1.2, 1.4, and 1.5, we shall use the planarity of J_{18} and Grinberg's Criterion [69] stating that if a plane graph has a hamiltonian cycle such that there are f_i faces of size i in the exterior of the cycle and f'_i faces of size i in the interior of the cycle, then

$$\sigma := \sum_i (i-2)(f_i - f'_i) = 0.$$

Let us call a face of a plane graph an *i-face* if dividing its size by 3, we obtain the remainder i (thus $i \in \{0, 1, 2\}$). Now we are ready to verify Properties 1.2, 1.4, and 1.5. By symmetry, for these it suffices to show that

1. there is no hamiltonian ad-path in J_{18} (Property 1.2),
2. there is no hamiltonian bc-path in $J_{18} - a$ (Property 1.5),
3. there is no hamiltonian bd-path in $J_{18} - a$ (Property 1.5),
4. there is no hamiltonian cd-path in $J_{18} - a$ (Property 1.5),
5. there exist no ad- and bc-path partitioning $V(J_{18})$ (Property 1.4), and
6. there exist no ac- and bd-path partitioning $V(J_{18})$ (Property 1.4).

To prove point 1, let us assume to the contrary that there exists a hamiltonian ad-path $\mathfrak{p}$ in J_{18}. Then we have a hamiltonian cycle $\mathfrak{h} := \mathfrak{p} + ad$ in the plane graph $J_{18} + ad$, which is a so-called Grinbergian graph (see e.g. [85], [141]), because all but one of its faces are 2-faces. These graphs are easily seen to be non-hamiltonian: let f_i and f'_i be as in Grinberg's theorem, using $\mathfrak{h}$ as the hamiltonian cycle; now σ is not divisible by 3, thus it cannot be 0, a contradiction.

Now we prove points 2, 3, and 4. The plane graph $J_{18} - a$ has only 2-faces and by adding any (one) of the edges bc, bd, cd we obtain a graph with one 0-face

and one 1-face that are on the opposite sides of the new edge, while all the other faces are still 2-faces. If there is a hamiltonian bc-path $\mathfrak{p}'$ in $J_{18} - a$, then $\mathfrak{h}' := \mathfrak{p}' + bc$ is a hamiltonian cycle in $(J_{18} - a) + bc$. Let f_i and f_i' be as in Grinberg's theorem again, using $\mathfrak{h}'$ as the hamiltonian cycle. The 0-face and the 1-face must be on different sides of $\mathfrak{h}'$, since they both contain the edge $bc \in E(\mathfrak{h}')$, thus σ is again indivisible by 3, a contradiction. The proof is the same for points 3 and 4.

Let us now prove point 5. Assume to the contrary that there exists the disjoint union $\mathfrak{u}$ of an ad- and a bc-path spanning J_{18}. Then $\mathfrak{h}'' := \mathfrak{u} + ab + cd$ is a hamiltonian cycle of the plane graph $J_{18} + ab + cd$, which has two 0-faces and all other faces are 2-faces, one of which is an octagon. Let f_i and f_i' be as in Grinberg's theorem, using $\mathfrak{h}''$ as the hamiltonian cycle. Then both 0-faces are on a different side of $\mathfrak{h}''$ than the octagon, since the octagon contains both ab and cd and these edges lie in $\mathfrak{h}''$. So the 0-faces are on the same side of $\mathfrak{h}''$, whence σ is again indivisible by 3, a contradiction.

Finally, we prove point 6. It is easy to see that the graph J_{18}' obtained by adding a new vertex v and the edges $ab, bc, cd, da, va, vb, vc, vd$ to J_{18} is also a planar graph. Now assume to the contrary that there exists the disjoint union of an ac- and a bd-path in J_{18}. Then J_{18}' would contain a subdivision of K_5 (consider the vertices a, b, c, d, v), a contradiction by Kuratowski's Theorem. □

Lemma 2.3. *The quintuple (J_{18}, a, b, c, d) of Figure 2.1 is both a K_1- and a K_2-cell.*

Proof. We implemented a straightforward computer program which we used to verify that the quintuple (J_{18}, a, b, c, d) indeed has the desired properties and thus is both a K_1-cell and a K_2-cell. The source code of this program can be found on GitHub [56] and in Section A.2 of the Appendix we included drawings so these results can also be verified by hand.

In order to show that (J_{18}, a, b, c, d) is a K_1-cell, by Lemma 2.2 it suffices to prove that if by deleting any inner vertex of J_{18} we obtain a graph in which at least one of the bad pairs of Properties 1.2, 1.4, 1.5(a) becomes good. Figure A.3 in the Appendix shows all inner vertex-deleted subgraphs of J_{18} up to symmetry and highlights a new good pair. Below each figure we mention which previously bad pair became good in this subgraph and to which property of suitable cells the bad pair belonged.

We now prove that (J_{18}, a, b, c, d) is a K_2-cell. It suffices to show that Properties 2.1–2.5 hold. Figure A.4 illustrates the proof of these properties. This figure shows all subgraphs of J_{18} up to symmetry obtained by deleting a pair of adjacent inner vertices and highlights a new good pair, i.e. all subgraphs

relating to Property 2.1. If this new good pair belonged to Property 1.5, we show the subgraph in which we also deleted the appropriate outer vertex. The figure also shows the proof of Property 2.2, i.e. all subgraphs in which we deleted a and one of its neighbours, together with a hamiltonian path highlighting the new good pair. Properties 2.3–2.5 are symmetric. □

From Theorem 2.1 and Lemma 2.3 we can immediately infer the following result.

Corollary 2.4. *There exists an infinite family $\mathcal{G}$ of non-hamiltonian planar K_1-hamiltonian K_2-hamiltonian graphs.*

The smallest member of $\mathcal{G}$, which we will call G_{48}, is depicted in Figure 2.2.

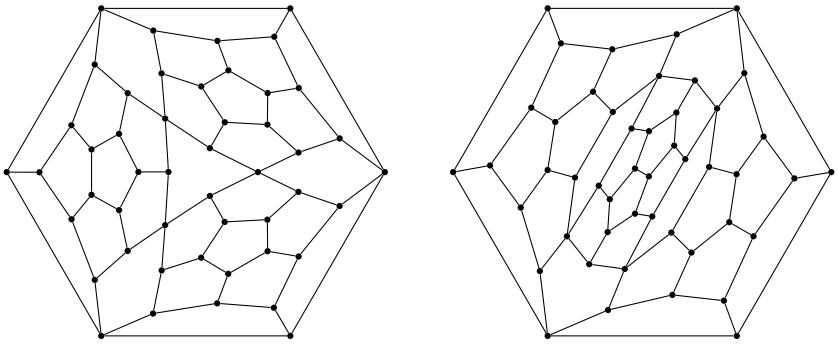


Figure 2.2: Two depictions of the planar hypohamiltonian K_2-hamiltonian graph G_{48} on 48 vertices obtained from three copies of J_{18}.

We provide a short alternative proof of Corollary 2.4 based only on the hypohamiltonicity and K_2-hypohamiltonicity of G_{48} (which could be determined easily using a computer, without the framework laid out in Theorem 2.1 and Lemma 2.3). We require the following definition, a result of Thomassen, and a lemma from [143]. In a 2-connected non-hamiltonian graph G, we call $\mathrm{exc}(G) \subset V(G)$, which contains every vertex v of G such that $G - v$ is non-hamiltonian, the set of *exceptional vertices* of G.

Lemma 2.5 (Thomassen [128])**.** *Let F_1 and F_2 be disjoint 3-fragments of hypohamiltonian graphs, not both trivial, with F_i having attachments X_i. Then $(F_1, X_1)\,(F_2, X_2)$ is hypohamiltonian.*

Lemma 2.6 ([143])**.** *Let G_1 and G_2 be disjoint K_2-hypohamiltonian graphs. For $i \in \{1, 2\}$, let G_i contain a 3-cut X_i and X_i-fragments F_i and F_i' such that*

for each $x \in X_i$ there is a hamiltonian path in $F_i - x$ and in $F_i' - x$ between the two vertices of $X_i - x$. This is fulfilled e.g. when X_i is non-trivial, or $\operatorname{exc}(G_i) \cap X_i = \emptyset$. *Then, if both F_1 and F_2 are non-trivial, or both F_i and F_{3-i}' are trivial, $(F_1, X_1)\,(F_2, X_2)$ is K_2-hypohamiltonian.*

Second proof of Corollary 2.4. Since G_{48} is hypohamiltonian, $\operatorname{exc}(G_{48}) = \emptyset$. Consider a cubic vertex v in G_{48} and let F and F' denote two copies of the non-trivial $N(v)$-fragment of G_{48} isomorphic to $G_{48} - v$. Let X (X') be the 3-cut in G_{48} corresponding to $N(v)$ in F (F'). By Lemma 2.6, $G := (F, X)\,(F', X')$ is K_2-hypohamiltonian; here we make use of the fact that, since G_{48} is hypohamiltonian, Lemma 2.5 implies that G is hypohamiltonian so $\operatorname{exc}(G) = \emptyset$. By construction G contains many cubic vertices, so we may iterate this procedure ad infinitum. □

Now that we know of infinitely many planar K_2-hypohamiltonian graphs, one can ask how quickly the number of pairwise non-isomorphic such graphs grows as a function of the graphs' order. Indeed, even leaving planarity aside, no non-trivial lower bound was known hitherto. Using the aforementioned lemmas, we can prove the following.

Theorem 2.7. *The number of planar hypohamiltonian K_2-hamiltonian graphs grows at least exponentially.*

Proof. Let A and B be the two graphs shown in Figure 2.3. They are non-isomorphic, of the same order, each obtained by removing two cubic vertices v, w from the planar hypohamiltonian K_2-hamiltonian graph G_{52} shown in Figure 2.5 from Section 2.3.3, where v and w lie on the boundary of the infinite face and are at distance 3. Consider

$$G := K_{1,3}\, F_1\, F_2\, \ldots\, F_k\, K_{1,3},$$

where each F_i is A or B, the operation is performed such that the corresponding copies of $N(v)$ and $N(w)$ are identified, and in the case of $K_{1,3}$, its three leaves are used for the identification. We abbreviate this convention by (†).

Since A and B as well as $K_{1,3}$ are planar, it is clear that the identifications behind the application of can be performed such that G is also planar. We now show that G is hypohamiltonian and K_2-hypohamiltonian. By (†), we will suppress mentioning explicitly the triples of vertices to which the identifications behind are applied, in order to improve readability. Clearly, since both $K_{1,3}\, F_1$ and $F_2\, K_{1,3}$ are vertex-deleted subgraphs of G_{52}, which itself is a hypohamiltonian K_2-hamiltonian graph, by Lemmas 2.5 and 2.6, the graph $K_{1,3}\, F_1\, F_2\, K_{1,3}$ is hypohamiltonian and K_2-hypohamiltonian. Using the same

argument, applying to $K_{1,3}\, F_1\, F_2$ and $F_3\, K_{1,3}$ yields the hypohamiltonian K_2-hamiltonian graph $K_{1,3}\, F_1\, F_2\, F_3\, K_{1,3}$. Iterating this procedure yields that G must be hypohamiltonian and K_2-hypohamiltonian.

Now we show that if

$$G_1 := K_{1,3}\, F_1^1\, F_2^1\, \ldots\, F_k^1\, K_{1,3} \qquad \text{and} \qquad G_2 := K_{1,3}\, F_1^2\, F_2^2 : \ldots\, F_k^2\, K_{1,3}$$

are isomorphic, then we either have $F_i^1 \cong F_i^2$ for all $i = 1, \ldots, k$ or $F_i^1 \cong F_{k-i+1}^2$ for all $i = 1, \ldots, k$, yielding an exponential number of pairwise distinct G's.

G_{52} has exactly one non-cubic vertex. For a non-trivial 3-cut to exist, we need at least three such vertices (see Proposition 1 (ii) of [143]). By construction and the fact that G_{52} has only trivial 3-cuts, G has exactly $k-1$ non-trivial 3-cuts, namely the ones between the F_i's. Let f be an isomorphism that maps G_1 onto G_2 and denote the 3-cut between F_i^1 and F_{i+1}^1 by X_i and the 3-cut between F_i^2 and F_{i+1}^2 by Y_i. For a subset Z of vertices of G_1 let us denote by $f(Z)$ the set of vertices onto which the vertices of Z are mapped by f. It is obvious that $f(X_1) = Y_i$ for some i. Moreover, the possible values for i are only 1 and k, since the deletion of the vertices of Y_i must give a two component graph with components of 48 and $47(k-1)+1$ vertices. If $i = 1$, then $f(X_2) = Y_2$, because the deletion of the vertices of $f(X_1) = Y_1$ and $f(X_2)$ must give a three component graph with components of 48, 44, and $47(k-2)+1$ vertices. The proof that $f(X_j) = Y_j$ in the $i = 1$ case is similar. Now it is obvious that $F_i^1 \cong F_i^2$ for all $i = 1, \ldots, k$. If $i = k$, then the same argument shows that $F_i^1 \cong F_{k-i+1}^2$ for all $i = 1, \ldots, k$, finishing the proof. □

We note that this also gives an alternative proof of Collier and Schmeichel's theorem stating that the number of hypohamiltonian graphs grows at least exponentially [29]. (In fact, here we show that this even holds restricted to planar graphs.)

One might wonder if there are further, not necessarily planar suitable cells that are both K_1- and K_2-cells (which are preferably smaller than the quintuple (J_{18}, a, b, c, d)). A related question is whether there exist K_1-cells in which all inner vertices, i.e. all vertices except a, b, c, d, have degree at least 4. This would solve an old question of Thomassen, namely whether hypohamiltonian graphs of minimum degree at least 4 exist [128]. (We note that all K_2-hypohamiltonian graphs we shall encounter in the remainder of this text have minimum degree 3.) Using such cells might even allow to find 4-connected hypohamiltonian graphs, whose existence is another open question of Thomassen [128] and one of the most important unresolved problems related to hypohamiltonicity, and the behaviour of longest cycle intersections in general. We recall that infinitely many 4-connected non-hamiltonian graphs exist in which exactly one vertex

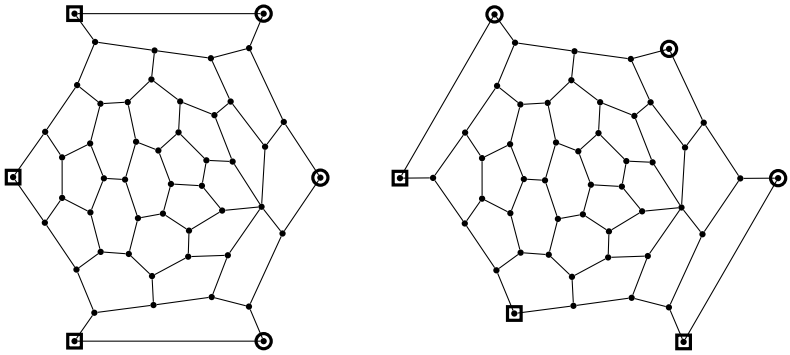


Figure 2.3: Graph A (left) and graph B (right) are two non-isomorphic graphs obtained from G_{52} by removing two cubic vertices. The neighbours of these removed vertices are indicated by squares (for the first vertex) and circles (for the second vertex).

has the property that its removal yields a non-hamiltonian graph [144], and that there are infinite families of planar hypohamiltonian graphs containing a constant number of cubic vertices (note that hypohamiltonian graphs have minimum degree at least 3), based on an operation of Thomassen [131, p. 41].

Using the generator `geng` [100], which can generate all connected (simple) graphs of a given order (and lower bound on the girth), in combination with our program from the proof of Lemma 2.3, we searched for other suitable cells, K_1-cells, as well as K_2-cells other than (J_{18}, a, b, c, d). The results of these computations are summarised in the following observations and Table 2.1. The suitable cells reported in this table up to 18 vertices can be obtained from the database of interesting graphs from the *House of Graphs* [30] by searching for the keywords "suitable cell".

Observation 2.8. *There are no suitable cells of order at most* 11 *and none of girth at least* 4 *of order* 12 *and* 13*. The smallest suitable cells of girth at least* 5 *have order* 16 *and their exact counts are listed in Table 2.1.*

Order	≤ 15	16	17	18	19
Number of cells	0	4	28	365	6 861

Table 2.1: Counts of all suitable cells of girth at least 5 of a given order up to 19 vertices.

Observation 2.9. *There is exactly one suitable cell of girth at least* 5 *up to* 19 *vertices which is a K_1-cell and exactly one of those suitable cells is a K_2-cell. In both cases it is the same cell, namely (J_{18}, a, b, c, d) from Figure 2.1.*

Corollary 2.10. *The quintuple (J_{18}, a, b, c, d) from Figure 2.1 is the smallest K_1- and K_2-cell of girth at least* 5*. Moreover, any other K_1- or K_2-cell of girth at least* 5 *must have at least* 20 *vertices.*

2.3 Orders

In this section we determine all K_2-hypohamiltonian graphs up to a given order for certain important classes of graphs and characterise the orders for which they exist. In particular, in Sections 2.3.1–2.3.4, we deal with general graphs, cubic graphs (including snarks), planar graphs, and cubic planar graphs, respectively. These results were obtained by using a combination of computational methods and theoretical arguments.

For the computational part, we amongst others implemented an efficient algorithm to test if a given graph is K_2-hypohamiltonian. The core of our algorithm is a straightforward procedure which searches for hamiltonian cycles in a graph. Starting with a path of two edges centered at a vertex of minimum degree with a chosen end, we recursively grow a path by adding a vertex to this end in each possible way. Given a path P with fewer than n vertices, if the first vertex of P is not adjacent to a vertex not in P, or there is a vertex not in P that is adjacent to fewer than two vertices that are not interior vertices of P, then P cannot be part of a hamiltonian cycle so we discontinue extending it. If P has n vertices and the ends are adjacent, we have found a hamiltonian cycle.

This algorithm for finding hamiltonian cycles can then be used to determine whether graphs are hamiltonian, K_1-hamiltonian, and K_2-hamiltonian.

The source code of this algorithm can be downloaded from GitHub [56] and in the subsections below we describe how we verified the correctness of our implementation.

2.3.1 The general case

A characterisation of the orders for which K_2-hypohamiltonian graphs exist is a K_2-analogue of a natural question—*For which n do there exist hypohamiltonian graphs on n vertices?*—addressed in a series of articles, among which the paper by Aldred, McKay, and Wormald [2] completely settles the problem by proving

that there is no hypohamiltonian graph on 17 vertices. Thus, the answer to the aforementioned question is that there exists a hypohamiltonian graph of order n if and only if $n \in \{10, 13, 15, 16\}$ or $n \geq 18$ is an integer.

The first author determined that all (seven) hypohamiltonian graphs on at most 16 vertices are K_2-hamiltonian, while the last author showed that the generalised Petersen graph $\mathrm{GP}(2, 11)$ is K_2-hamiltonian [143] (that $\mathrm{GP}(2, 11)$ is hypohamiltonian was shown by Bondy [13]). Combining these graphs via Lemmas 2.5 and 2.6, we obtain hypohamiltonian K_2-hamiltonian graphs of order 10, 13, 15, 16, 18, and every integer which is at least 20; a detailed account is given in the proof of Theorem 2.11. For all other (non-trivial) orders it was an open question whether a K_2-hypohamiltonian graph of that order exists. It is not difficult to verify that K_2-hypohamiltonian graphs are 3-connected. Hence, their minimum degree is at least 3.

We used the program `geng` [100] to generate all connected simple graphs of a given order (with a given lower bound on the girth and minimum degree) and then filtered these graphs using our program to check whether they are K_2-hypohamiltonian or not. The results of these computations are summarised in Table A.1 in Section A.3 of the Appendix. We verified all results for orders 5–12, 14–16 (with girth at least 4), and 17–19 (with girth at least 5) using an independent program. Checking higher orders with the independent program would be unfeasible without allocating a lot of computational resources. Previously, all non-hamiltonian graphs up to 13 vertices were already determined in [48]. The K_2-hypohamiltonian graphs from Table A.1 can be downloaded from the database of interesting graphs from the *House of Graphs* [30] by searching for the keywords "K2-hypohamiltonian".

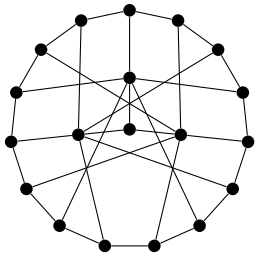
Theorem 2.11. *The Petersen graph is the only K_2-hypohamiltonian graph of order at most 12, any K_2-hypohamiltonian graph of order 14 (if such a graph exists) must have girth 3, and any K_2-hypohamiltonian graph of order 17 (if such a graph exists) must have girth at most 4. Moreover, for any integer $n \in \{10, 13, 15, 16\}$ or $n \geq 18$, there exists a K_2-hypohamiltonian graph of order n.*

Proof. The statements from the theorem's first sentence follow directly from our computations summarised in Table A.1 in the Appendix. It also gives an independent verification of the first author's computations that there exists an n-vertex hypohamiltonian K_2-hamiltonian graph for every $n \in \{10, 13, 15, 16\}$.

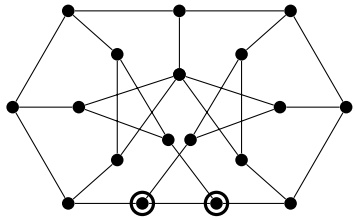
As shown by the last author in [143], the generalised Petersen graph $\mathrm{GP}(2, 11)$ on 22 vertices is a cubic hypohamiltonian K_2-hamiltonian graph. Since each of the aforementioned K_2-hamiltonian graphs of orders 10, 13, 15, and 16 is also hypohamiltonian, and each has at least five cubic vertices, we can apply

Lemmas 2.5 and 2.6 to obtain hypohamiltonian K_2-hamiltonian graphs of orders 18, 20, and 21, each containing a cubic vertex. Here we always apply the operation to two 3-fragments of hypohamiltonian graphs obtained by removing a cubic vertex of the graph. Thus, when applying , in each of the underlying hypohamiltonian graphs at most four cubic vertices are removed or rendered non-cubic. Furthermore, note that if the underlying graphs have order p and q, respectively, so that the fragments have order $p-1$ and $q-1$, respectively, the graph resulting from the application of has order $p+q-5$.

Our computations yield the existence of a hypohamiltonian K_2-hamiltonian graph on 19 vertices, see Figure 2.4a. Hence, there exist hypohamiltonian K_2-hamiltonian graphs of every order between 18 and 22, each containing a cubic vertex. Combining these graphs, via Lemma 2.6, with the Petersen graph a suitable number of times, we obtain a K_2-hypohamiltonian graph of every order that is at least 18. □



(a) A hypohamiltonian K_2-hamiltonian graph on 19 vertices.



(b) The K_2-hypohamiltonian graph of smallest order which has girth ≥ 5 and is not hypohamiltonian. It has order 18. The exceptional vertices are marked.

Figure 2.4: Two K_2-hypohamiltonian graphs.

We emphasise that the only integers n for which it is not yet settled whether K_2-hypohamiltonian graphs of order n exist are 14 and 17; we leave this as an open problem.

We end this section with a brief remark, which can be directly inferred from our computations, on the order of the smallest non-hypohamiltonian K_2-hypohamiltonian graph.

Observation 2.12. *The smallest K_2-hypohamiltonian graph that contains a vertex whose deletion yields a non-hamiltonian graph has order at least* 14 *and at most* 18.

The upper bound of 18 is given by the graph from Figure 2.4b. It is the smallest K_2-hypohamiltonian graph of girth at least 5 which is not hypohamiltonian.

2.3.2 The cubic case

General cubic graphs

Our main motivation for characterising the orders for which cubic K_2-hypohamiltonian graphs exist is our goal to give a K_2-analogue of Thomassen's Theorem 5.1 from [131]; it characterises all orders for which cubic hypohamiltonian graphs exist. It is elementary to see that a cubic K_2-hypohamiltonian graph must have girth at least 5. Using the program `snarkhunter` [19], we generated all cubic graphs of girth at least 5 up to 32 vertices and verified which of them are K_2-hypohamiltonian using the program mentioned at the beginning of Section 2.3. Our findings are summarised in Table 2.2. All results up to and including $n = 26$ were verified by an independently written program. The K_2-hypohamiltonian graphs from Table 2.2 can be downloaded from the database of interesting graphs from the *House of Graphs* [30] by searching for the keywords "cubic K2-hypohamiltonian". The counts for cubic hypohamiltonian graphs were obtained by Aldred, McKay, and Wormald [2] and improved by the first and last author [59].

Order	Total	Non-ham.	K_2-hypoham.	hypo- and K_2-ham.
10	1	1	1	1
12–16	60	0	0	0
18	455	3	0	0
20	5 783	15	1	1
22	90 938	110	3	3
24	1 620 479	1 130	0	0
26	31 478 584	15 444	6	5
28	656 783 890	239 126	14	12
30	14 621 871 204	4 073 824	15	14
32	345 975 648 562	75 458 941	12	12

Table 2.2: Counts of all graphs of girth at least 5 that are cubic; cubic and non-hamiltonian; cubic and K_2-hypohamiltonian; cubic, hypohamiltonian, and K_2-hypohamiltonian, respectively. Recall that cubic K_2-hypohamiltonian graphs have girth at least 5.

By restricting the generation in `snarkhunter` [19] to cubic graphs of girth at least 6, we were also able to determine the following.

Observation 2.13. *Up to and including* 36 *vertices there are only two* K_2*-hypohamiltonian graphs of girth at least* 6*, namely the flower snark* J_7 *and the flower snark* J_9*.*

For the next theorem's proof, we will make use of the following definition and result from [143]. Consider a graph G containing a 5-cycle $C = v_0 \dots v_4 v_0$ such that every v_i is cubic. Denote the neighbour of v_i not on C by v_i'. Then the cycle C is called *extendable* if for any i, taking indices modulo 5, (i) there exists a hamiltonian cycle $\mathfrak{h}$ in $G - v_i$ with $v_{i-2}v_{i+2} \notin E(\mathfrak{h})$ and (ii) there exists a hamiltonian cycle $\mathfrak{h}'$ in $G - v_i'$ with $\mathfrak{h}' \cap C = v_{i-2}v_{i-1}v_iv_{i+1}v_{i+2}$.

Lemma 2.14 ([143])**.** *Let* G *be a* K_2*-hamiltonian graph containing an extendable* 5*-cycle. Then there exists a* K_2*-hamiltonian graph* G' *of order* $|V(G)| + 20$ *containing an extendable* 5*-cycle. If* G *is non-hamiltonian or cubic, then so is* G'*, respectively. If* G *has girth* 5*, then* G' *has girth* 5*. If* G *is plane and* C *a facial cycle in* G*, then* G' *is planar.*

Theorem 2.15. *There exists a cubic* K_2*-hypohamiltonian graph of order* n *if and only if* $n \in \{10, 20, 22\}$ *or* $n \geq 26$ *is even.*

Proof. It follows from the counts in Table 2.2 that there exist cubic K_2-hypohamiltonian graphs of order 10, 20, 22, 26, 28, 30, and 32, and that there exist no cubic K_2-hypohamiltonian graphs of order 12, 14, 16, 18, and 24. Hence, it remains to show that there exist cubic K_2-hypohamiltonian graphs of even orders 34 and upwards. In Table 2.3 in Subsection 2.3.2 we give cubic K_2-hypohamiltonian graphs on 34 and 36 vertices and Table 2.4 presents cubic K_2-hypohamiltonian graphs on 38, 40, and 44 vertices. We wrote a computer program which tests if a given graph has an extendable 5-cycle. Its implementation can be found on GitHub [56] and using this program we verified that there exist cubic K_2-hypohamiltonian graphs of order $n \in \{22, 26, 28, 30, 32, 34, 36, 38, 40, 44\}$ that contain an extendable 5-cycle. In the same GitHub repository we included a file with a cubic K_2-hypohamiltonian graph for each of these orders together with an extendable 5-cycle C and paths which prove that C is indeed an extendable 5-cycle so that this can be verified by hand. By Lemma 2.14 we get an n-vertex graph with an extendable 5-cycle for any even $n \geq 34$, which completes the proof. □

Snarks

It is common to call a graph a *snark* if it is cubic and cyclically 4-edge-connected, its chromatic index is 4, and its girth is at least 5. Snarks are particularly interesting since for several famous conjectures (such as Tutte's

5-flow Conjecture) it has been proven that if the conjecture is false, the smallest possible counterexample must be a snark. The interplay between snarks and hypohamiltonicity has been studied by various authors; we refer the reader to [60] for further references. Our aim in this section is to study the analogous problem for K_2-hypohamiltonicity.

In [18] Brinkmann, the first author, Hägglund, and Markström presented an algorithm to generate snarks, used it to generate all such graphs up to 36 vertices, and determined the number of hypohamiltonian snarks among them (and later this was extended up to order 38 for snarks of girth at least 6, see [17]). Using our program for testing K_2-hypohamiltonicity, we determined which of these snarks are K_2-hypohamiltonian. The results can be found in Table 2.3 and these graphs can also be downloaded from the *House of Graphs* [30] at `https://houseofgraphs.org/Snarks`. The counts up to and including $n = 30$ vertices were verified by an independently written program.

n	Girth	Total	hypoham.	K_2-hypoham.	hypo- and K_2-ham.
10	≥ 5	1	1	1	1
12–16	≥ 5	0	0	0	0
18	≥ 5	2	2	0	0
20	≥ 5	6	1	1	1
22	≥ 5	20	2	2	2
24	≥ 5	38	0	0	0
26	≥ 5	280	95	6	5
28	≥ 5	2 900	31	14	12
30	≥ 5	28 399	104	9	9
32	≥ 5	293 059	13	11	11
34	≥ 5	3 833 587	31 198	1 036	936
36	≥ 5	60 167 732	10 838	3 849	3008
38	≥ 6	39	29	20	10

Table 2.3: Counts of all snarks, hypohamiltonian snarks, K_2-hypohamiltonian snarks, and snarks which are both hypohamiltonian and K_2-hypohamiltonian.

In [60] the first and last author determined all hypohamiltonian snarks with at least 38 and at most 44 vertices which can be obtained by taking the dot product on two smaller hypohamiltonian snarks. We verified which of these hypohamiltonian snarks are K_2-hypohamiltonian and the resulting counts can be found in Table 2.4. (We needed these graphs for the proof of Theorem 2.15.)

The technique from Lemma 2.14 does not preserve the chromatic index of a graph. We will therefore use the dot product to create larger graphs whilst preserving 3-regularity, chromatic index, and K_2-hypohamiltonicity by imposing

Order	d.p. hypoham.	d.p. hypo- and K_2-ham.
38	51 431	2 482
40	8 820	2 522
42	20 575 458	450 886
44	8 242 146	1 606 786

Table 2.4: Counts of hypohamiltonian snarks which can be obtained by taking the dot product on two smaller hypohamiltonian snarks (d.p. hypoham.) and the number of K_2-hamiltonian snarks among them (d.p. hypo- and K_2-ham.).

constraints on the smaller graphs. We recall the definition of the dot product as was given by the last author in [143].

Let G and H be disjoint graphs on at least six vertices. For independent edges ab, cd in G and adjacent cubic vertices x and y in H, consider $G' := G - ab - cd$ and $H' := H - x - y$, and let a', b' be the neighbours of x in $H - y$ and c', d' be the neighbours of y in $H - x$. Then the *dot product* $G \cdot H$ is defined as the graph

$$(V(G) \cup V(H'), E(G') \cup E(H') \cup \{aa', bb', cc', dd'\}).$$

A similar result concerning K_2-hypohamiltonian graphs and the dot product can already be found in [143]. As mentioned by the author in [143], this result is quite impractical due to the many requirements imposed on G and H. It also contained certain inaccuracies, which we rectify in Section 2.4. The following theorem gives different constraints on G and H, which make it more practical to use.

In a graph G, consider pairwise distinct vertices s_1, s_2, t_1, t_2. We will refer to a pair of paths $(\mathfrak{p}, \mathfrak{q})$ as *disjoint spanning s_1t_1- and s_2t_2-paths* if $\mathfrak{p}$ is an s_1t_1-path, $\mathfrak{q}$ is an s_2t_2-path, $\mathfrak{p}$ and $\mathfrak{q}$ are disjoint, and each vertex of G lies in either $\mathfrak{p}$ or $\mathfrak{q}$.

Theorem 2.16. *Let G and H be disjoint non-hamiltonian graphs with $a, b, c, d \in V(G)$ and a', b', c', d', x, $y \in V(H)$ as introduced in the definition of the dot product—in particular, x and y are cubic— $a, b \notin N_G(c) \cup N_G(d)$, $a' \notin N_H(b')$, and $c' \notin N_H(d')$. The graph $G \cdot H$ is non-hamiltonian and K_2-hamiltonian if*

(i) *for any $vw \in E(G')$ there exists in $G - v - w$ a hamiltonian ab-path not containing cd or a hamiltonian cd-path not containing ab or disjoint spanning s_1t_1- and s_2t_2-paths containing neither ab, nor cd, where $\{s_1, s_2\} = \{a, b\}$ and $\{t_1, t_2\} = \{c, d\}$;*

(ii) *for any* $s \in \{a, b\}$ *and* $t \in \{c, d\}$*, the graph* G *admits a hamiltonian* st*-path, containing neither* ab*, nor* cd *and* G *admits disjoint spanning* ab*- and* cd*-paths, containing neither* ab*, nor* cd*;*

(iii) $G - a$ *and* $G - b$ *contain a hamiltonian cycle through* cd*, and* $G - c$ *and* $G - d$ *contain a hamiltonian cycle through* ab*;*

(iv) $H - x$ *and* $H - y$ *are hamiltonian and there exist in* H' *disjoint* $a'b'$*- and* $c'd'$*-paths spanning* $V(H')$*;*

(v) *for any* $vw \in E(H)$ *with* $v, w \notin \{x, y\}$ *there exists in* $H - x - y - v - w$ *a hamiltonian* $s't'$*-path with* $s' \in \{a', b'\}$ *and* $t' \in \{c', d'\}$ *or disjoint spanning* $s'_1t'_1$*- and* $s'_2t'_2$*−paths with* $\{s'_1, s'_2\} = \{a', b'\}$ *and* $\{t'_1, t'_2\} = \{c', d'\}$*; and*

(vi) $H - x - a'$*,* $H - x - b'$*,* $H - y - c'$*, and* $H - y - d'$ *are hamiltonian.*

Proof. Since G and H are non-hamiltonian their dot product will also be non-hamiltonian. This was shown for example in [143]. We show $G \cdot H$ is K_2-hamiltonian. Consider $vw \in E(G')$. Suppose that by (i) there exists a hamiltonian ab-path in $G - v - w$ not containing cd, call it $\mathfrak{p}$. By (iv), $H - x - y$ contains a hamiltonian $a'b'$-path $\mathfrak{q}$. Now $\mathfrak{p} \cup \mathfrak{q}$ is a hamiltonian cycle in $G \cdot H - v - w$. The proof if $G - v - w$ has a hamiltonian cd-path not containing ab is analogous. For the final case of (i), assume that $G - v - w$ has disjoint spanning s_1t_1- and s_2t_2-paths $\mathfrak{p}$ and $\mathfrak{q}$ not containing ab and cd, where $\{s_1, s_2\} = \{a, b\}$ and $\{t_1, t_2\} = \{c, d\}$. By (iv) there exist disjoint spanning ab- and cd-paths $\mathfrak{p}'$ and $\mathfrak{q}'$ in $H - x - y$. Then $\mathfrak{p} \cup \mathfrak{p}' \cup \mathfrak{q} \cup \mathfrak{q}'$ is a hamiltonian cycle in $G \cdot H - v - w$.

By (iii) there is a hamiltonian cd-path $\mathfrak{p}$ in $G - a$, and (vi) implies that there is a hamiltonian $c'd'$-path $\mathfrak{q}$ in $H - x - y - a'$. Hence, $\mathfrak{p} \cup \mathfrak{q}$ is a hamiltonian cycle in $G \cdot H - a - a'$. The cases when removing b, b' or c, c' or d, d' can be dealt with in a similar way.

Consider $vw \in E(H)$ with $v, w \notin \{x, y\}$. Suppose we are in the first case of (v) and there exists in $H - x - y - v - w$ a hamiltonian $s't'$-path $\mathfrak{p}$ with $s' \in \{a', b'\}$ and $t' \in \{c', d'\}$. Let $N_{G \cdot H}(s') \cap V(G) = \{s\}$ and $N_{G \cdot H}(t') \cap V(G) = \{t\}$. By (ii) there exists a hamiltonian st-path containing neither ab, nor cd, which together with $\mathfrak{p}$ forms a hamiltonian cycle in $G \cdot H - v - w$. Finally, suppose we are in the second case of (v) and that $H - x - y - v - w$ contains disjoint spanning $s'_1t'_1$- and $s'_2t'_2$-paths $\mathfrak{p}'$ and $\mathfrak{q}'$, where $\{s'_1, s'_2\} = \{a', b'\}$ and $\{t'_1, t'_2\} = \{c', d'\}$. By (ii) there exist disjoint spanning ab- and cd-paths $\mathfrak{p}$ and $\mathfrak{q}$ containing neither ab, nor cd in G. Then $\mathfrak{p} \cup \mathfrak{p}' \cup \mathfrak{q} \cup \mathfrak{q}'$ is a hamiltonian cycle in $G \cdot H - v - w$. □

It was proven by Isaacs in [81] that the dot product of two snarks is again a snark. Hence, the dot product of two K_2-hypohamiltonian snarks G and

H satisfying the properties of the previous theorem is a K_2-hypohamiltonian snark. The following lemma will help us iteratively apply the dot product to K_2-hypohamiltonian snarks.

Lemma 2.17. *Let G and H be defined as in Theorem 2.16 such that $a, b, c, d \in V(G)$ satisfy (i)–(iii) and $x, y \in V(H)$ satisfy (iv)–(vi). Let $x_G y_G \in E(G)$ such that x_G and y_G are cubic vertices and such that $N_G[x_G] \cup N_G[y_G]$ does not intersect $\{a, b, c, d\}$. Let a'_G, b'_G be the neighbours of x_G not equal to y_G and let c'_G, d'_G be the neighbours of y_G not equal to x_G. If*

- *$G - x_G$ contains a hamiltonian ab-path not containing cd or a hamiltonian cd-path not containing ab;*
- *$G - y_G$ contains a hamiltonian ab-path not containing cd or a hamiltonian cd-path not containing ab; and*
- *G contains disjoint spanning ab- and cd-paths, one containing x_G and the other y_G,*

then $x_G, y_G \in V(G \cdot H)$ satisfy properties (iv)–(vi).

Proof. We first show (iv). Suppose $G - x_G$ contains a hamiltonian ab-path $\mathfrak{p}$ not containing cd. Since H satisfies (iv), there is a hamiltonian $a'b'$-path $\mathfrak{q}$ in $H - x - y$. Then $\mathfrak{p} \cup \mathfrak{q}$ is a hamiltonian cycle in $G \cdot H - x_G$. If $G - x_G$ contains a hamiltonian cd-path not containing ab, a similar argument shows $G \cdot H - x_G$ is hamiltonian. The same reasoning proves that $G \cdot H - y_G$ is hamiltonian.

Assume without loss of generality that G contains disjoint spanning ab- and cd-paths $\mathfrak{p}$ and $\mathfrak{q}$ such that $x_G \in V(\mathfrak{p})$ and $y_G \in V(\mathfrak{q})$. Since H satisfies (iv) we get disjoint spanning $a'b'$- and $c'd'$-paths $\mathfrak{p}'$ and $\mathfrak{q}'$. Then $G \cdot H$ gets spanned by two disjoint cycles $\mathfrak{p} \cup \mathfrak{p}'$ and $\mathfrak{q} \cup \mathfrak{q}'$, the former containing x_G and the latter containing y_G. Hence, we get disjoint spanning $a'_G b'_G$- and $c'_G d'_G$-paths in $G \cdot H - x_G - y_G$. This shows $x_G, y_G \in V(G \cdot H)$ satisfy (iv).

We show (v). Let $vw \in E(G \cdot H)$ with $v, w \notin \{x, y\}$. Suppose $G \cdot H - v - w$ contains a hamiltonian cycle. If this cycle contains $x_G y_G$, then $G \cdot H - x_G - y_G - v - w$ admits a hamiltonian $s't'$-path with $s' \in \{a'_G, b'_G\}$ and $t' \in \{c'_G, d'_G\}$. If the cycle does not contain $x_G y_G$, it contains $a'_G x_G b'_G$ and $c'_G y_G d'_G$, which in $G \cdot H - x_G - y_G - v - w$ leads to disjoint spanning $s'_1 t'_1$- and $s'_2 t'_2$-paths with $\{s'_1, s'_2\} = \{a'_G, b'_G\}$ and $\{t'_1, t'_2\} = \{c'_G, d'_G\}$. We need to show that $G \cdot H - v - w$ contains a hamiltonian cycle.

Suppose that $vw \in E(G)$ and assume that there exists in $G - v - w$ a hamiltonian ab-path $\mathfrak{p}$ not containing cd. Since H satisfies (iv) there is a hamiltonian $a'b'$-path $\mathfrak{q}$ in $H - x - y$. Then $\mathfrak{p} \cup \mathfrak{q}$ is a hamiltonian cycle in $G \cdot H - v - w$.

Similarly, if there exists in $G - v - w$ a hamiltonian cd-path not containing ab, there is a hamiltonian cycle in $G \cdot H - v - w$. Otherwise, suppose $G - v - w$ contains disjoint spanning s_1t_1- and s_2t_2-paths $\mathfrak{p}$ and $\mathfrak{q}$ containing neither ab, nor cd, where $\{s_1, s_2\} = \{a, b\}$ and $\{t_1, t_2\} = \{c, d\}$. Since H satisfies (iv) there are disjoint spanning $a'b'$- and $c'd'$-paths $\mathfrak{p}'$ and $\mathfrak{q}'$. Then, $\mathfrak{p} \cup \mathfrak{p}' \cup \mathfrak{q} \cup \mathfrak{q}'$ is a hamiltonian cycle in $G \cdot H - v - w$.

Suppose that $vw \in E(H)$ and assume that there exists in $H - x - y - v - w$ a hamiltonian $s't'$-path $\mathfrak{q}$ with $s' \in \{a', b'\}$ and $t' \in \{c', d'\}$. Since G satisfies (ii), there is a hamiltonian st-path $\mathfrak{p}$ in G with $s \in N_{G \cdot H}(s') \cap V(G)$ and $t \in N_{G \cdot H}(t') \cap V(G)$. We get a hamiltonian cycle $\mathfrak{p} \cup \mathfrak{q}$ in $G \cdot H - v - w$. If $H - x - y - v - w$ contains disjoint spanning $s'_1t'_1$- and $s'_2t'_2$-paths $\mathfrak{p}'$ and $\mathfrak{q}'$, with $\{s'_1, s'_2\} = \{a', b'\}$ and $\{t'_1, t'_2\} = \{c', d'\}$. Since G satisfies (ii) it admits disjoint spanning ab- and cd-paths $\mathfrak{p}$ and $\mathfrak{q}$, containing neither ab, nor cd. Hence, $\mathfrak{p} \cup \mathfrak{p}' \cup \mathfrak{q} \cup \mathfrak{q}'$ is a hamiltonian cycle in $G \cdot H - v - w$.

Finally, assume without loss of generality that $vw = aa'$. Since G satisfies (ii), there is a hamiltonian cd-path $\mathfrak{p}$ in $G - a$. Since H satisfies (vi) there is a hamiltonian $c'd'$-path $\mathfrak{p}'$ in $H - x - y - a'$. Then $\mathfrak{p} \cup \mathfrak{p}'$ is a hamiltonian cycle in $G \cdot H - a - a'$. The same argument works for bb', cc', and dd'. Hence, $x_G, y_G \in V(G \cdot H)$ satisfy (v).

Since $G \cdot H$ is K_2-hamiltonian by Theorem 2.16 condition (vi) is also satisfied. □

We can now classify for which orders there exist K_2-hypohamiltonian snarks.

Theorem 2.18. *There exists a K_2-hypohamiltonian snark of order n if and only if $n \in \{10, 20, 22\}$ or $n \geq 26$ is even.*

Proof. It follows from the counts in Table 2.3, that there exists no K_2-hypohamiltonian snark of order 12, 14, 16, 18, and 24 and that the Petersen graph is a K_2-hypohamiltonian snark of order 10. Using our computer program, we verified that there exist K_2-hypohamiltonian snarks of order $n \in \{20, 22, 26, 28, 30, 32, 34, 36, 44\}$ satisfying (iv)–(vi) for some pair of adjacent vertices. Recall that we found K_2-hypohamiltonian snarks of order 44, see Table 2.4. In particular, the flower snark J_5 satisfies these conditions, as well as (i)–(iii), and the conditions of Lemma 2.17 for some pair of independent edges and pair of adjacent vertices. In the GitHub repository [56], we attached a file containing certificates which prove the claims above for a K_2-hypohamiltonian snark of each of these orders so that one can verify them by hand.

Let G be the flower snark J_5 and H be a graph satisfying (iv)–(vi) for some pair of adjacent vertices. By Lemma 2.17, $G \cdot H$ satisfies conditions (iv)–(vi) and has order $|V(G)| + |V(H)| - 2 = |V(H)| + 18$. We can then take the dot

product with G in this way ad infinitum. Taking the dot product of the flower snark J_5 with the graphs mentioned above, we can find a K_2-hypohamiltonian snark for all even orders $n \geq 26$. □

Note that the above also gives us an alternative proof for Theorem 2.15.

2.3.3 The planar case

In [78] Holton and Sheehan asked for the order of the smallest planar hypohamiltonian graph. This problem remains open, but we know that the answer is at least 23 and at most 40, see [59] and [85], respectively. The latter paper also proves that there exist planar hypohamiltonian graphs of order n for every $n \geq 42$ and that the smallest planar hypohamiltonian graph of girth 5 has order 45. It is natural to raise the same questions for K_2-hypohamiltonian graphs.

Using the program `plantri` [21] we generated all cyclically 4-edge-connected planar graphs of girth 5 up to 48 vertices, as well as an incomplete sample of such graphs for larger orders, and verified which of them are K_2-hypohamiltonian using the program mentioned at the beginning of Section 2.3. The results are summarised in Table 2.5. Note that a K_2-hypohamiltonian graph is always cyclically 4-edge-connected [143]. As another correctness test, we stored all non-hamiltonian graphs obtained from our computation with `plantri`, confirmed their non-hamiltonicity, and tested which of them are K_2-hamiltonian using an independent program. The K_2-hypohamiltonian graphs from Table 2.5 can be obtained from the database of interesting graphs from the *House of Graphs* [30] by searching for the keywords "planar K2-hypohamiltonian". The unique planar K_2-hypohamiltonian graph of girth 5 on 48 vertices is the graph G_{48} from Figure 2.2. We also determined that only two of the K_2-hypohamiltonian graphs from Table 2.5 are hypohamiltonian: next to G_{48}, the graph from Figure 2.5 is also hypohamiltonian.

Order	≤ 47	48	49	50	51	52	53
Number of graphs	0	1	≥ 9	≥ 11	≥ 0	≥ 6	≥ 9

Table 2.5: Counts of K_2-hypohamiltonian graphs among planar graphs of girth 5.

Using `plantri` we also generated all 3-connected planar graphs up to and including 17 vertices. We verified using our program that none of these are K_2-hypohamiltonian. As K_2-hamiltonian graphs are 3-connected, this gives us a lower bound for the order of the smallest planar K_2-hypohamiltonian graph.

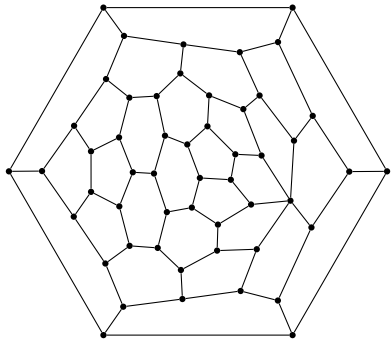


Figure 2.5: A planar hypohamiltonian K_2-hamiltonian graph of girth 5 on 52 vertices.

Observation 2.19. *For planar K_2-hypohamiltonian graphs the following statements hold.*

(i) There exist no planar K_2-hypohamiltonian graphs on at most 17 vertices.

(ii) The smallest planar K_2-hypohamiltonian graph of girth 5 has order 48. There is exactly one such graph of that order: the graph G_{48} from Figure 2.2.

(iii) There exist planar K_2-hypohamiltonian graphs of order $48, 49, 50, 52, 53$.

(iv) There exist planar hypohamiltonian K_2-hamiltonian graphs of order 48 and 52.

Holton and Sheehan [78] asked whether there is a number n_0 such that for every integer $n \geq n_0$ there exists a planar hypohamiltonian graph on n vertices. This was solved, affirmatively, by Araya and the third author [142]. In a similar vein, Holton [77] writes "*Suppose the smallest planar hypohamiltonian graph had n_0 vertices. Does there exist a planar hypohamiltonian graph on n vertices for all $n > n_0$ or are there "gaps"?*" For an infinite graph family $\mathcal{F}$ we call such a gap a *Holton gap*; more specifically, if there exist integers a, b, c with $a < b < c$ such that there exist members of $\mathcal{F}$ of order a and c, but not of b, we say that $\mathcal{F}$ has a *Holton gap at b*.

Holton's problem has not yet been settled—in particular, we do not know whether planar hypohamiltonian graphs have a Holton gap—but substantial progress has been made; for the state-of-the-art, see [85]. We recall that therein it is shown that there exist planar hypohamiltonian graphs of order 40 as well

as of order n for every $n \geq 42$, but it remains an open question whether planar hypohamiltonian graphs have a Holton gap at 41. We now discuss K_2-analogues of these questions.

Theorem 2.20. *There exists a planar K_2-hypohamiltonian graph of order n for every integer $n \geq 177$.*

Proof. In Figure 2.6, we present planar K_2-hypohamiltonian graphs of order n for every $n \in \{48, 49, 50, 52, 53\}$. By Lemma 2.6, these five graphs yield the existence of planar K_2-hypohamiltonian graphs of order n for every $n \in \{177, \ldots, 196\}$.

Applying Lemma 2.6 a suitable number of times to the graphs in Figure 2.6, we get a planar K_2-hypohamiltonian graph of order n for every $n \in \{177, \ldots, 196\}$. We show that by choosing appropriate 3-cuts these graphs contain an extendable 5-cycle.

Let G_1 and G_2 be the graphs to which we apply the operation of Lemma 2.6. For $i \in \{1, 2\}$, let G_i contain a 3-cut X_i and X_i-fragments F_i and F_i' such that the conditions of the lemma are satisfied and such that F_i is non-trivial. We show that an extendable 5-cycle $C = v_0 \ldots v_4$ in G_1 disjoint from X_1 leads to an extendable 5-cycle in $(F_1, X_1)\,(F_2, X_2) =: G$. Indeed, for every $i = 0, \ldots, 4$, we find a hamiltonian cycle in $G_1 - v_i$, which leads to a hamiltonian path in F_1 with endpoints in X_1. Since there exists, for each $x \in X_2$, a hamiltonian path in $F_2 - x$ between the two vertices of $X_2 - x$, it follows that there exists a hamiltonian cycle in $G - v_i$. Similarly, we have for every $i = 0, \ldots, 4$ a hamiltonian cycle in $G_1 - v_i'$, where v_i' is the neighbour of v_i that does not lie on C. Noting that v_i' lies in F_1, we see in the same way that there exists a hamiltonian cycle in $G - v_i'$. Hence, G contains an extendable 5-cycle. It is easy to see that any trivial 3-cut in G_1 which lies in F_1 gives rise to a trivial 3-cut in G and by Theorem 6 in [144], we can infer that a vertex in G is exceptional if and only if it was exceptional in G_1 or G_2.

Figure 2.6 depicts K_2-hypohamiltonian planar graphs of order 48, 49, 50, 52, and 53, their exceptional vertices, and an extendable 5-cycle. In the GitHub repository [56], we attached a file with these planar K_2-hypohamiltonian graphs together with an extendable 5-cycle C and paths which prove that C is indeed an extendable 5-cycle so that this can be verified by hand. We can then apply the operation of Lemma 2.6 to the graphs of Figure 2.6 a suitable number of times with the appropriate 3-cuts to obtain planar K_2-hypohamiltonian graphs of order n with an extendable 5-cycle for every $n \in \{177, \ldots, 196\}$.

The proof is now completed by applying Lemma 2.14 to these twenty graphs. □

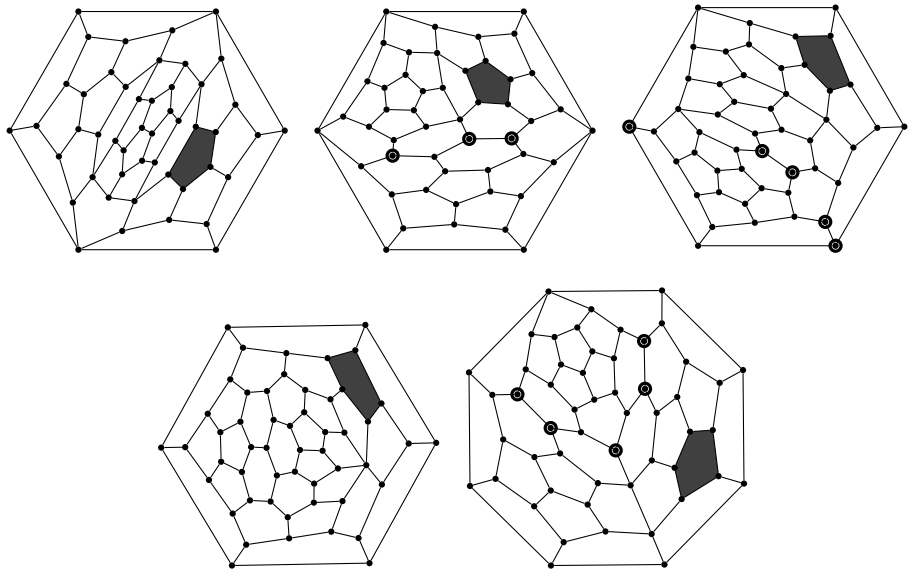


Figure 2.6: Planar K_2-hypohamiltonian graphs on 48, 49, 50, 52, and 53 vertices together with an extendable 5-cycle and all of their exceptional vertices.

2.3.4 The cubic planar case

Using the program `plantri` [21] we generated all cyclically 4-edge-connected cubic planar graphs of girth 5 up to 78 vertices and verified which of them are K_2-hypohamiltonian using the program mentioned at the beginning of Section 2.3. The results are summarised in Table 2.6.

The counts of all cyclically 4-edge-connected non-hamiltonian cubic planar graphs of girth 5 up to 76 vertices were already tabulated before by McKay in [98] and in [59] the first and last author determined that there are exactly 860 350 such graphs on 78 vertices. All cyclically 4-edge-connected non-hamiltonian cubic planar graphs of girth 5 were stored and we verified that these are indeed non-hamiltonian and contain the same number of K_2-hypohamiltonian graphs as in Table 2.6 using an independent program.

The three cubic planar K_2-hypohamiltonian graphs up to 74 vertices are shown in Figure 2.7 and the six remaining cubic planar K_2-hypohamiltonian graphs up to 78 vertices are shown in Figure A.5 in the Appendix. The K_2-hypohamiltonian graphs from Table 2.6 can be obtained from the database of interesting graphs from the *House of Graphs* [30] by searching for the keywords "cubic planar K2-hypohamiltonian".

Order	≤ 66	68	70	72	74	76	78
Number of graphs	0	1	1	0	1	3	3

Table 2.6: Counts of all K_2-hypohamiltonian graphs among cubic planar graphs up to 78 vertices.

The following observation immediately follows from the results in Table 2.6. (Recall that K_2-hypohamiltonian graphs are 3-connected and cyclically 4-edge-connected. Cubic K_2-hypohamiltonian graphs must have girth at least 5. Since a planar 3-connected graph can have girth at most 5, cubic planar K_2-hypohamiltonian graphs must have girth 5.)

Observation 2.21. *There exists exactly one cubic planar K_2-hypohamiltonian graph of order $68, 70$, and 74, and no such graph on fewer than 68 vertices or of order 72. Moreover, there exist such graphs of order 76 and 78.*

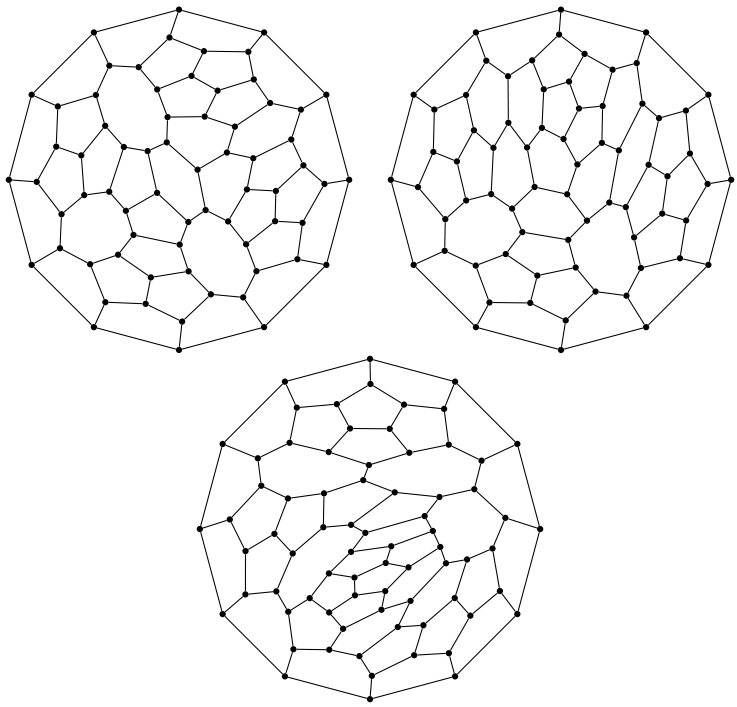


Figure 2.7: All cubic planar K_2-hypohamiltonian graphs on 68, 70, and 74 vertices, respectively.

Corollary 2.22. *The family of cubic planar K_2-hypohamiltonian graphs has a Holton gap at* 72.

Araya and the third author [7] discovered a cubic planar hypohamiltonian graph of order 70 (a smaller such graph, if it exists, must have at least 54 vertices [59]), and it was shown subsequently that such graphs of order n exist for every even $n \geq 74$, see [144]. Thus, one is tempted to believe that the family of cubic planar hypohamiltonian graphs *also* has a Holton gap at 72, but this remains unknown.

Theorem 2.23. *There exist infinitely many cubic planar K_2-hypohamiltonian graphs.*

Proof. In Figure A.6 in the Appendix we provide a 68-vertex cubic planar K_2-hypohamiltonian graph G containing an extendable cycle; the proof is contained in the aforementioned figure. By Lemma 2.14, the proof is complete. □

Finally, we point out that among all cubic planar K_2-hypohamiltonian graphs that we were able to investigate, exceptional vertices always occurred (i.e. we do not know whether cubic planar hypohamiltonian K_2-hamiltonian graphs exist):

Observation 2.24. *All cubic planar K_2-hypohamiltonian graphs up to and including* 78 *vertices have at least two exceptional vertices, and there exists a cubic planar K_2-hypohamiltonian graph of order* 76 *containing exactly five exceptional vertices.*

2.4 Concluding remarks

1. Errata. In the last author's paper [143], the first part of this series of articles, there is one minor issue and one more significant issue. We now explain these omissions and present corrections.

The minor issue concerns Lemma 3 in Section 3.1 of [143]. Let $\mathcal{H}_k$ be the set of all 2-connected n-vertex graphs of circumference $n-1$ that contain exactly k exceptional vertices. The aforementioned lemma states: "Let i, j be non-negative integers and consider disjoint graphs $G \in \mathcal{H}_i$ and $H \in \mathcal{H}_j$. We require G and H to contain cubic vertices x and y, respectively, such that $N(x) \cap \mathrm{exc}(G) = \emptyset$ and $N(y) \cap \mathrm{exc}(H) = \emptyset$. Let F_G (F_H) be the non-trivial $N(x)$-fragment ($N(y)$-fragment) of G (H). Then $(F_G, N(x))$ $(F_H, N(y)) \in \mathcal{H}_{i+j}$."

The issue here is that if x or y are exceptional, the statement is false. So in the lemma's second sentence, $N(x)$ must be replaced by $N[x]$, i.e. the closed

neighbourhood of x, and $N(y)$ by $N[y]$. After performing this correction, we obtain precisely the statement of Theorem 6 from the last author's paper [144].

In [143], Lemma 3 is used in the proof of Theorem 1, whose statement remains valid. The only change that must be made in the proof of Theorem 1 is that in its last paragraph, the sentence "By construction, Γ_k contains a cubic vertex y_k such that no vertex of $N(y_k)$ is exceptional." must be replaced with: By construction, Γ_k contains a cubic vertex y_k such that no vertex of $N[y_k]$ is exceptional.

The major issue concerns Proposition 6 in Section 3.2 of [143], which discusses the dot product in the context of K_2-hamiltonicity. Unfortunately, the third and fourth paragraph of its "proof" contain errors. In order to correct these, we need to change, in the proposition's statement, condition (v) to (v)' and add a sixth condition as described below:

(v)' For any $vw \in E(H)$ with $v, w \notin \{x, y\}$ there exists in $H - x - y - v - w$ a Hamiltonian st-path with $s \in \{a', b'\}$ and $t \in \{c', d'\}$.

(vi) $H - x - a'$, $H - x - b'$, $H - y - c'$, and $H - y - d'$ are hamiltonian.

The third (and penultimate) paragraph of the proposition's "proof" can now be saved as we get a hamiltonian cycle in $G - a - a'$ since there now exists a hamiltonian $c'd'$-path in $H - x - y - a'$ by condition (vi); one proceeds analogously for the hamiltonicity of $G - b - b'$, $G - c - c'$, and $G - d - d'$. In the fourth paragraph of the proposition's "proof" it is stated that by (v) there exists in $H - v - w$ a hamiltonian $s't'$-path $\mathfrak{p}$ with $s' \in \{a', b'\}$ and $t' \in \{c', d'\}$. The issue is that such a path may start in a', visit x, y, d' (in this order), then continue through H to b', and, again through H, end in c'. Such a path cannot be extended to the desired cycle by the arguments given in the "proof". By replacing condition (v) with (v'), such a path might still exist, but we are now guaranteed to have a path which *can* be extended, as described in [143], to the desired cycle.

2. Bipartite graphs. Hypohamiltonian graphs cannot be bipartite. However, it follows from an operation of Thomassen [131, p. 41] that for any $\varepsilon > 0$ there exists a bipartite graph H and a hypohamiltonian graph G such that H is an induced subgraph of G and $|V(H)|/|V(G)| > 1 - \varepsilon$. By an operation of the last author [143], this statement is also true for K_2-hypohamiltonian graphs. Grötschel asked whether there is a bipartite graph admitting no hamiltonian path, but in which every vertex-deleted subgraph does contain a hamiltonian path, see [68, Problem 4.56]. Related to this, Ozeki [105] asked whether there is a non-hamiltonian bipartite graph with bipartition (X, Y) such that $G - x - y$ is hamiltonian for all $x \in X$ and all $y \in Y$.

Here we address, computationally, a relaxation of this problem in tune with

the article's main theme, K_2-hamiltonicity. (Note that every bipartite K_2-hamiltonian graph must be balanced.) We found no K_2-hypohamiltonian bipartite graphs up to and including order 16, and no K_2-hypohamiltonian bipartite graphs of girth at least 6 up to and including order 25. (None of the other graphs we had already generated in Table A.1 were bipartite.) Recently, Brinkmann, McKay, and the first author [20] showed that the smallest 3-connected non-hamiltonian cubic bipartite graph has 50 vertices and they also generated a sample of 238 531 such graphs up to 64 vertices. We verified that none of these graphs are K_2-hypohamiltonian. Among planar 3-connected bipartite graphs there are no K_2-hypohamiltonian graphs up to and including 32 vertices. Barnette conjectured that every 3-connected cubic planar bipartite graph is hamiltonian and this conjecture was verified up to 90 vertices in [20], so there are no 3-connected cubic planar K_2-hypohamiltonian bipartite graphs up to at least 90 vertices.

3. Toward Grünbaum's conjecture. Going back to Grünbaum's problem, one goal is to find K_2-hypohamiltonian graphs with as few hamiltonian vertex-deleted subgraphs as possible. Let $\mathcal{G}_3$ be the family of all cubic K_2-hypohamiltonian graphs and

$$\rho_3 := \sup_{G \in \mathcal{G}_3} \frac{|\mathrm{exc}(G)|}{|V(G)|}.$$

In [143], the last author showed that $\rho_3 \geq \frac{1}{4}$. Any improvement of this bound would be very welcome.

4. Girth. It is natural to wonder what the smallest K_2-hypohamiltonian graph of a certain girth is. For an integer $k \geq 3$, let g_k denote the order of a smallest K_2-hypohamiltonian graph of girth k. Our computations imply that $g_3 \geq 14$ and $g_4 \geq 17$. We do not know whether K_2-hypohamiltonian graphs of girth 3 or 4 exist. The smallest K_2-hypohamiltonian graph of girth 5 is Petersen's, so $g_5 = 10$. There exists a K_2-hypohamiltonian graph of girth 6 and order 25—it is the smallest hypohamiltonian graph of girth 6, first described in [59]—and none on fewer vertices, so $g_6 = 25$. We have computed that $g_k \geq 26$ for all $k \geq 7$, but we do not know whether K_2-hypohamiltonian graphs of girth greater than 6 exist. The smallest cubic K_2-hypohamiltonian graph of girth 6 is the flower snark J_7. As mentioned in Theorem 2.9 of [59], the 28-vertex Coxeter graph is the only non-hamiltonian cubic graph with girth 7 up to at least 42 vertices (and Coxeter's graph is not K_2-hamiltonian), the smallest non-hamiltonian cubic graph with girth 8 has at least 50 vertices, and the smallest non-hamiltonian cubic graph with girth 9 has at least 66 vertices.

Acknowledgements

Several of the computations for this work were carried out using the supercomputer infrastructure provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government. The research of Jan Goedgebeur and Jarne Renders was supported by Internal Funds of KU Leuven. The research of Gábor Wiener was supported by the National Research, Development and Innovation Office (NKFIH), Hungary, Grant no. K-124171 and is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 funding scheme. The research of Carol T. Zamfirescu was supported by a Postdoctoral Fellowship of the Research Foundation Flanders (FWO).

Chapter 3

Generation and New Infinite Families of K_2-hypohamiltonian Graphs

This chapter is a minimally edited version of the published work.

J. Goedgebeur, J. Renders and C. T. Zamfirescu. Generation and new infinite families of K_2-hypohamiltonian graphs. *Discret. Math.*, 347(7):113981, 2024. DOI: 10.1016/j.disc.2024.113981

My contributions to this paper, in accordance with the contributor role taxonomy (CRediT)[1], consist of: Conceptualisation, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review and Editing, and Visualisation.

3.1 Introduction

A graph is called *hypohamiltonian* if it is non-hamiltonian and the removal of any vertex leads to a hamiltonian graph. We call a graph K_2-*hamiltonian* if the removal of any pair of adjacent vertices leads to a hamiltonian graph. If it is

[1] See https://credit.niso.org/.

also non-hamiltonian, it is a K_2-*hypohamiltonian* graph. In this article we will study K_2-hypohamiltonian graphs. We continue work from [53]; for motivation of these concepts, we refer to the introduction and references given in a paper by the third author [143].

Historically, in the hypohamiltonian case, it was of great interest to investigate the existence of hypohamiltonian graphs for a given order. For instance, in 1966 Herz, Duby and Vigué used a computer search [75] to prove that there are no hypohamiltonian graphs of orders 11 and 12, and in 1967 Lindgren showed the existence of an infinite family of hypohamiltonian graphs [93]. Aldred, McKay and Wormald completed the characterisation of the orders for which hypohamiltonian graphs exist in 1997 [2], demonstrating the absence of a hypohamiltonian graph of order 17. This proved that there exists a hypohamiltonian graph of order n if and only if $n \in \{10, 13, 15, 16\}$, or n is an integer greater than or equal to 18. A historical overview including the remaining orders was given by Holton and Sheehan in [78].

In [53], using mathematical constructions as well as a computational filter method, we showed that the Petersen graph is the smallest K_2-hypohamiltonian graph. Furthermore, we were able to prove the existence of K_2-hypohamiltonian graphs on orders $10, 13, 15, 16$, and from 18 onwards, as well as their non-existence up to order 9 and for orders 11 and 12. However, whether such graphs exist for orders 14 and 17 remained unknown, and the question of the existence of K_2-hypohamiltonian graphs with girth 3 and 4 persisted.

To address these questions, we describe a specialised generator that exhaustively generates all pairwise non-isomorphic K_2-hypohamiltonian graphs of a given order. We used this to determine whether or not K_2-hypohamiltonian graphs exist on these two open orders, to increase the known lower bounds on the order of the smallest K_2-hypohamiltonian graphs with girth 3 or girth 4 and to exhaustively determine which graphs are K_2-hypohamiltonian up to higher orders. We did this by proving that there are several *obstructions* for K_2-hypohamiltonicity, i.e. properties that cannot hold for K_2-hypohamiltonian graphs. We then used these obstructions as bounding criteria in the generation algorithm. This method is inspired by a generator for hypohamiltonian graphs showcased in [59], which was later also extended for generating *platypus graphs* [63] (i.e. non-hamiltonian graphs for which every vertex-deleted subgraph is traceable).

Currently, the order of a smallest planar K_2-hypohamiltonian graph is not known. This is also the case for the smallest planar hypohamiltonian graph. However, results by Aldred, McKay and Wormald [2] implied a lower bound of 18 for the latter. This was later sharpened to 23 in 2017 [59]. Restricting our generator to the class of planar graphs we increase the lower bound from [53]

for the order of a smallest planar K_2-hypohamiltonian graph from 18 to 24. The smallest known planar K_2-hypohamiltonian graph was given in the same paper and has 48 vertices.

A hypohamiltonian graph cannot be bipartite. However, a question by Grötschel [68, Problem 4.56] asked whether there exist bipartite *hypotraceable graphs*, i.e. graphs for which every vertex-deleted subgraph has a hamiltonian path, but which do not contain one themselves. We study here the existence of bipartite K_2-hypohamiltonian graphs. In [53] a lower bound of 17 on the order of a smallest bipartite K_2-hypohamiltonian graph was given. Similarly as in the planar case, by restricting the generation to bipartite graphs we increase this lower bound to 30. However, whether bipartite K_2-hypohamiltonian graphs exist remains unknown.

In Section 3.2 we present the exhaustive generation algorithm and the obstructions for K_2-hypohamiltonian graphs and prove the correctness of the generator. In Section 3.3 we introduce new ways of constructing infinite families of K_2-hypohamiltonian graphs. In particular, we introduce an operation for the construction of K_2-hypohamiltonian graphs, which are not hypohamiltonian, and which preserve planarity under certain conditions. We use this to classify the orders for which planar K_2-hypohamiltonian graphs exist, building on work from [53].

We also describe a new infinite family of hypohamiltonian K_2-hamiltonian graphs. Members of this family with n vertices have a maximum degree of approximately $n/3$ and approximately $2n$ edges. It is of interest what the maximum degree and maximum number of edges in a K_2-hypohamiltonian graph can be, as was historically the case for hypohamiltonian graphs. For the hypohamiltonian case this question was answered by Thomassen [131] by describing a family of hypohamiltonian graphs attaining up to some constant the upper bounds on the maximum degree and size, which are $(n-3)/2$ and $n^2/4$ for a graph of order n, respectively. However, in the K_2-hypohamiltonian case no non-trivial upper bounds for the maximum degree and size are known.

Throughout this paper we assume all graphs to be simple, finite and connected unless mentioned otherwise. K_2-hypohamiltonian graphs are 3-connected and hence have minimum degree at least 3. We will use this fact tacitly throughout the paper. A path with end-vertex v is a *v-path*, and a v-path with end-vertex $w \neq v$ is a *vw-path*. We denote the complement of a graph G by G^c, and by $N_G[v]$ the subset of $V(G)$ containing v and all its neighbours in a graph G. We will write $N[v]$ if it is clear we are referring to a subset of $V(G)$.

3.2 Generating K_2-hypohamiltonian graphs

In this section, we present our algorithm to generate all pairwise non-isomorphic K_2-hypohamiltonian graphs of a given order. The fundamental idea behind the algorithm is based on the generator for hypohamiltonian graphs from [59], which in turn is based on work by Aldred, McKay and Wormald [2]. However, we introduce essential new bounding criteria specifically designed for K_2-hypohamiltonian graphs which are vital for the efficiency of the algorithm and adapt the algorithm from [59] for use in the K_2-hypohamiltonian case.

3.2.1 Obstructions for K_2-hypohamiltonicity

We will start this section by presenting several lemmas on properties that hold true for K_2-(hypo)hamiltonian graphs. A selection of these will be used as bounding criteria in the algorithm to reduce the search space, i.e. if a graph does not satisfy one of the properties, it cannot be K_2-(hypo)hamiltonian. The selection of which obstructions or bounding criteria we apply is based on an experimental analysis as there is usually a trade-off between how quickly these conditions can be computed and how much of the search space they reduce.

In [2] Aldred, McKay and Wormald used what they called type A, B and C obstructions. These are specific configurations that cannot occur in hypohamiltonian graphs. We will use a similar terminology, but note however that here they indicate obstructions for K_2-hypohamiltonicity instead of hypohamiltonicity. The statements of the obstructions and lemmas contain some subtle differences compared to the hypohamiltonian case. Moreover, we will introduce several lemmas specifically for K_2-hypohamiltonian graphs. While some of these are adaptations from [2] and [59], we also introduce some new ones. The precise definition of these obstructions for K_2-hypohamiltonicity will be given later in this section.

A *non-trivial partition* (W, X) of a set V is a pair of two disjoint subsets W and X of V such that their union is V and neither is empty. Let G be a possibly disconnected graph. We denote by $p(G)$ the minimum number of pairwise disjoint paths needed to cover all the vertices of G, by $V_1(G)$ all vertices of degree 1 in G and by $I(G)$ the set of all isolated vertices and all isolated K_2's

of G, i.e. isolated edges with their endpoints. Define

$$k(G) = \begin{cases} 0 & \text{if } G \text{ is empty,} \\ \max\left\{1, \left\lceil \frac{|V_1|}{2} \right\rceil\right\} & \text{if } I(G) = \emptyset \text{ but } G \text{ is not empty,} \\ |I(G)| + k(G - I(G)) & \text{else.} \end{cases}$$

Lemma 3.1 (Aldred, McKay, Wormald [2])**.** *For a graph G, we have $k(G) \leq p(G)$.*

Let $\mathfrak{p}$ be a path or cycle in G and S a subset of the vertices of G. We define $c_{\mathfrak{p}}|_S$ to be the number of components of $\mathfrak{p}$ restricted to $G[S]$.

Lemma 3.2. *Let G be a K_2-hamiltonian graph and (W, X) a non-trivial partition of $V(G)$. If $G[X]$ contains adjacent vertices and $|X| \geq 3$, then*

$$p(G[W]) < |X| - 1 \quad \textit{and} \quad k(G[W]) < |X| - 1.$$

Proof. Let v and w be adjacent vertices of $G[X]$. Since G is K_2-hamiltonian, there is a hamiltonian cycle $\mathfrak{h}$ in $G - v - w$. Then $c_{\mathfrak{h}}|_W$ must be at least $p(G[W])$, hence $c_{\mathfrak{h}}|_{X-v-w}$ must also be at least $p(G[W])$, since $c_{\mathfrak{h}}|_W = c_{\mathfrak{h}}|_{X-v-w}$. As $c_{\mathfrak{h}}|_{X-v-w}$ is at most $|X| - 2$, we get

$$p(G[W]) \leq c_{\mathfrak{h}}|_{X-v-w} \leq |X| - 2 < |X| - 1.$$

By Lemma 3.1 the result follows. □

Consider a graph G containing a non-trivial partition (W, X) of the vertices of G such that X has at least three vertices and contains a pair of adjacent vertices. If $p(G[W]) \geq |X| - 1$, we call (W, X) a *type A obstruction* and if $k(G[W]) \geq |X| - 1$, we call (W, X) a *type B obstruction*. For the efficiency of our algorithm, we only consider type A obstructions where $G[W]$ is a union of disjoint paths with the extra condition that none of these paths in $G[W]$ have adjacent endpoints. In this way computing $p(G[W])$ is equivalent to counting the degree 1 vertices of $G[W]$, dividing this number by two and adding the number of degree 0 vertices.

The case when X does not contain adjacent vertices might be of theoretical interest. We explore this in the next lemma.

Lemma 3.3. *Let G be a K_2-hamiltonian graph and (W, X) a non-trivial partition of $V(G)$ with $|X| > 1$. If $G[X]$ does not contain adjacent vertices, then*

$$p(G[W - w]) < |X| \quad \textit{and} \quad k(G[W - w]) < |X|$$

for any $w \in W$ adjacent to a vertex in X.

Proof. Let v, w be adjacent vertices of G such that v is in X and w in W. We can find such vertices since G is connected. As G is K_2-hamiltonian, we have a hamiltonian cycle $\mathfrak{h}$ in $G - v - w$. If W contains only one element, we are done. Assume otherwise; then the number of components of $\mathfrak{h}$ restricted to W is at least $p(G[W - w])$. Hence, the number of components of $\mathfrak{h}$ restricted to X is at least $p(G[W - w])$. Since this number of components is at most $|X| - 1$, we are done by Lemma 3.1. □

Note that adding a vertex to a subgraph of G can only increase the number of pairwise disjoint paths needed to cover that subgraph by at most one. This yields the following corollary.

Corollary 3.4. *Under the conditions of Lemma 3.3, we have $p(G[W]) < |X|+1$.*

Lemma 3.5. *Let G be a K_2-hamiltonian graph and (W, X) a non-trivial partition of the vertices of G, such that W is an independent set. Suppose $G[X]$ contains adjacent vertices v and w. Let n_1 and n_2 be the number of vertices of $X - v - w$ joined to one or more than one vertex of W, respectively. Then $2n_2 + n_1 \geq 2|W|$.*

Proof. Let $\mathfrak{h}$ be a hamiltonian cycle in $G - v - w$. Since W is an independent set, the number of edges of $\mathfrak{h}$ between W and X must be $2|W|$. On the other hand X can supply at most $2n_2 + n_1$ edges. The result follows. □

Let G be a graph which is not K_2-hamiltonian and suppose that for a non-trivial partition (W, X) of the vertices of G the remaining assumptions of Lemma 3.5 are met, but $2n_2 + n_1 < 2|W|$ for some adjacent $v, w \in X$. Then we call (W, X, vw) a *type C obstruction.*

Lemma 3.6. *Let G be a K_2-hypohamiltonian graph containing a triangle uvw. Then u has at least two neighbours not in $N[v] \cup N[w]$.*

Proof. Suppose that u has at most one neighbour not in $N[v] \cup N[w]$. There exists a hamiltonian cycle in $G - v - w$. Note that this cycle does not use the edge ux, otherwise replacing ux with $uvwx$, we would obtain a hamiltonian cycle in G. Hence, it must contain an edge ux where x is a neighbour of u that also neighbours either v or w in G. However, we can then replace this edge by $xvwu$ or $xwvu$ which yields a hamiltonian cycle in G, which is a contradiction. □

Let G be a graph and u a vertex of G lying on a triangle uvw such that u has at most one neighbour not in $N[v] \cup N[w]$. Then we say (u, uvw) is a *triangle obstruction* in G.

Corollary 3.7 ([143])**.** *Let G be a K_2-hypohamiltonian graph. The vertices of a triangle in G have degree at least* 4*.*

Lemma 3.8. *Let G be a K_2-hypohamiltonian graph containing a* 4*-cycle $uvwx$. Then u has at least two neighbours not in $N[w]$.*

Proof. Assume that u has at most one neighbour not in $N[w]$. There exists a hamiltonian cycle in $G - v - w$ and it must contain the edge uy where y is a neighbour of u adjacent to w in G. Replacing this edge with $ywvu$ yields a hamiltonian cycle in G, which is a contradiction. □

Suppose that u belongs to a 4-cycle $uvwx$ such that u has at most one neighbour which is not w and which is non-adjacent to w. Then we say $(u, uvwx)$ is a 4*-cycle obstruction*.

Corollary 3.9 ([143])**.** *Let G be a K_2-hypohamiltonian graph. The vertices of a* 4*-cycle in G have degree at least* 4*.*

A *diamond* is a K_4 from which we remove one edge. Its *central edge* is the edge between the two cubic vertices.

Corollary 3.10. *Let G be a K_2-hypohamiltonian graph containing a diamond with vertices a, b, c, d and central edge ac. Then the degrees of a and c (in G) are at least* 5*.*

Let G be a non-hamiltonian graph. If adding a non-edge e to G makes G hamiltonian, we call e a *bad edge*. Fix an obstruction O in G. We say a non-edge e *destroys* O in G if $G + e$ does not contain O. Sometimes it is necessary to add multiple non-edges of G in order to destroy an obstruction O. We say for each of these non-edges that they *work towards the destruction* of O. We will call such a non-edge of G a *good edge*. We will more formally define good edges for the different obstructions.

Let G' be a K_2-hypohamiltonian graph such that G is a spanning subgraph of G'. Let G contain a type A obstruction (W, X), i.e. (W, X) is a non-trivial partition of $V(G)$, $G[X]$ contains at least three vertices some of which are adjacent and $p(G[W]) \geq |X| - 1$. G' cannot contain a type A obstruction by Lemma 3.2 since G' is K_2-hypohamiltonian, hence there exists an edge in $E(G') \setminus E(G)$ with endpoints in different components of $G[W]$. We will call such an edge a *good* (W, X) *A-edge*. Similarly, let G contain a type B obstruction (W, X), i.e. (W, X)

is a non-trivial partition of $V(G)$, $G[X]$ contains at least three vertices some of which are adjacent and $k(G[W]) \geq |X| - 1$. By definition of $k(G)$, we can only destroy a type B obstruction by adding an edge which gives us fewer isolated vertices and isolated K_2's in $G[W]$ or which gives us fewer vertices of degree 1 in $G[W] - I(G[W])$. Hence, an edge in G^c is a *good* (W, X) *B-edge* if both of its endpoints lie in W and at least one of its endpoints has degree at most 1 in $G[W]$. It follows from Lemma 3.1 that every type B obstruction is a type A obstruction but not vice versa. However, experimental evidence showed that a type A obstruction is only very sporadically not a type B obstruction. Since the computational overhead of searching for type B obstructions is greater than that of searching for type A obstructions we obtained better results by only searching for type A obstructions. We therefore omit the type B obstructions entirely from Algorithm 2 below and our implementation.

Let G contain a type C obstruction (W, X, vw), i.e. (W, X) is a non-trivial partition, W and independent set, v and w are adjacent vertices in $G[X]$, n_1 and n_2 are the number of vertices of $X - v - w$ joined to one or more than one vertex, respectively, of W, and $2n_2 + n_1 < 2|W|$. A non-edge e of G destroys or works towards the destruction of (W, X, vw) if adding it to G increases $2n_2 + n_1$ or if W is not an independent set of $G + e$. Hence, we call an edge in G^c a *good* (W, X, vw)*-edge* if it connects a vertex of $X - v - w$ which has at most one neighbour in W to W or if both of its endpoints are in W.

If G contains a triangle obstruction (u, uvw), i.e. uvw is a triangle in G such that $|N(u) \setminus (N(v) \cup N(w))| \leq 1$, then there must be an edge $ux \in E(G') \setminus E(G)$ such that x is not an element of $N[v] \cup N[w]$. We call such an edge a *good* (u, uvw)*-edge*.

Similarly, if G contains a 4-cycle obstruction $(u, uvwx)$, i.e. $uvwx$ is a 4-cycle of G and $|N(u) \setminus N[w]| \leq 1$, then there must be an edge $uy \in E(G') \setminus E(G)$ such that y is not an element of $N[w]$. We call such an edge a *good* $(u, uvwx)$*-edge*.

3.2.2 Algorithm

The goal of our algorithm is to generate all pairwise non-isomorphic K_2-hypohamiltonian graphs of a given order n. Its pseudocode is split into two parts given by Algorithm 1 and Algorithm 2. The former contains the initialisation of the algorithm, which is only called once, while the latter contains the main routine, which recursively calls itself many times. We initialise the generation by taking an $(n-2)$-cycle and a copy of K_2, i.e. an isolated edge uv. In this way $G - u - v$ is hamiltonian. We then get new graphs by adding two edges with endpoint u and two edges with endpoint v to the graph in all possible ways that do not make the graph hamiltonian, filtering out isomorphic copies. Note that

any K_2-hypohamiltonian graph of order n must have a subgraph isomorphic to one of these new graphs, as K_2-hypohamiltonian graphs are 3-connected. Then on each of these new graphs we call Algorithm 2, which will recursively add edges until we can prune the search. We use obstructions to minimise the number of edges we need to add in each iteration. During both Algorithm 1 and Algorithm 2, we only add edges between existing vertices of the graph. Hence, if a graph is hamiltonian, we prune the search and do not examine its supergraphs. In particular, suppose we add an edge e to a non-hamiltonian graph G and obtain a hamiltonian graph. Then we forbid the adding of e to any graph containing G as a subgraph by keeping and dynamically updating a list of "forbidden edges" associated with the graph which is currently being constructed by the algorithm. For efficiency reasons, we store this list of forbidden edges as a bitvector.

Computing whether or not a graph is K_2-hypohamiltonian is computationally quite expensive as determining if a graph is hamiltonian is already NP-complete. Therefore, before doing this we look for obstructions that make it impossible for the graph to be K_2-hypohamiltonian and perform the K_2-hypohamiltonicity check in the few cases that the graph does not contain any obstructions. (E.g. for $n = 18$, only 98 of the 6 116 186 generated intermediate candidate graphs contained no obstructions. Three of these were K_2-hypohamiltonian.) Moreover, since we know that the parent graph was non-hamiltonian, if the current graph would be hamiltonian, the hamiltonian cycle must go through the most recently added edge. For the efficiency of the algorithm it is important that in each call of Algorithm 2 as few edges as possible are added. To this end, after finding an obstruction, we only need to add edges that destroy or work towards the destruction of this obstruction instead of adding all non-forbidden edges. In Theorem 3.11 we show this is sufficient. In Algorithm 2, we investigate which obstructions are present in the graph and count how many non-hamiltonian successors each investigated obstruction has. The obstruction with the fewest will be the one we destroy or work towards the destruction of. The good edges for this obstruction which, if added, will not yield a hamiltonian graph will be stored in the list `EdgesToBeAdded`. While checking the different obstructions, we will encounter edges that, if added, give a hamiltonian graph. We add these to the list `NewForbiddenEdges` as they should not be added to any supergraph of the one we are currently inspecting.

Adding edges would in many cases lead to graphs isomorphic to ones generated earlier. In order to prevent this, we keep a list of the canonical forms of all generated intermediate candidate graphs (computed by `nauty` [100]). Every time we inspect a candidate graph, its canonical form is compared to those in the list. If it is already present we can reject it, otherwise it is added.

Note that more advanced isomorphism rejection methods exist, for example the

Algorithm 1 Generate all K_2-hypohamiltonian graphs of order n

1: let $G := C_{n-2} + \{\{u, v\}, \{uv\}\}$
2: **for** every way of adding two edges incident to u and two edges incident to v **do**
3: Call the resulting graph G'
4: **if** G' is hamiltonian **then**
5: Go to the next iteration of the loop.
6: **if** G' is isomorphic to a previously constructed graph **then**
7: Go to the next iteration of the loop.
8: Create list of bitvectors `ForbiddenEdges` containing only edges which, if added, make G' hamiltonian
9: AddEdges(G', `ForbiddenEdges`) // i.e. perform Algorithm 2.

canonical construction path method by McKay [99]. However, such methods are not compatible with the abovementioned obstructions. Hence, this seemingly naive isomorphism rejection method is very suitable for this problem.

In the following theorem, we show that the algorithm produces all K_2-hypohamiltonian graphs of a given order.

Theorem 3.11. *Let n be a positive integer. When Algorithm 1 terminates, we have output precisely all pairwise non-isomorphic K_2-hypohamiltonian graphs of order n.*

Proof. By line 35 of Algorithm 2, we see that only K_2-hypohamiltonian graphs are output. By line 1 we see that no isomorphic copies are ever output. It remains to show that the algorithm produces all such pairwise non-isomorphic graphs of order n. Let G be a K_2-hypohamiltonian graph. It is easy to see that there is a spanning subgraph G_0 of G consisting of an $(n-2)$-cycle $\mathfrak{h}$ and a copy of K_2, i.e. an edge uv, disjoint from $\mathfrak{h}$ such that u is adjacent to two vertices of $\mathfrak{h}$ and v is adjacent to two vertices of $\mathfrak{h}$. Clearly if G_0 is hamiltonian, then G is as well, which is a contradiction.

We now proceed by induction and show that if we call Algorithm 2 on a graph which is isomorphic to a spanning subgraph of G on m edges with $|E(G_0)| \leq m < |E(G)|$, then we will call Algorithm 2 on a graph isomorphic to a spanning subgraph of G on $m+1$ edges. Assume that G' is such a graph with m edges for $|E(G_0)| \leq m < |E(G)|$ and that we call Algorithm 2 on G'. Without loss of generality, we assume that G' is a subgraph of G.

Suppose that the algorithm did not find any obstructions for G'. Since G' is a strict subgraph of G, there must be an edge e in $E(G) \setminus E(G')$. Clearly $G' + e$

Algorithm 2 AddEdges(Graph G, List `ForbiddenEdges`)

if an isomorphic copy of G was generated before **then**
  return
Create List `NewForbiddenEdges` = `ForbiddenEdges`
Create List `EdgesToBeAdded` containing all edges of G^c
for type A obstructions (W, X) **do**
  for all non-forbidden good (W, X) A-edges e **do**
    Add bad edges e to `NewForbiddenEdges`
  if #non-forbidden good (W, X) A-edges < #`EdgesToBeAdded` **then**
    `EdgesToBeAdded` = non-forbidden good (W, X) A-edges
for all type C obstructions (W, X) **do**
  for all non-forbidden good C-edges e **do**
    Add bad edges e to `NewForbiddenEdges`
  if #non-forbidden good (W, X, vw)-edges < #`EdgesToBeAdded` **then**
    `EdgesToBeAdded` = non-forbidden good (W, X, vw)-edges
for all vertices w of degree 2 **do**
  for all edges e in G^c having w as an endpoint **do**
    Add bad edges e to `NewForbiddenEdges`
  if #non-forbidden edges in G^c incident to w < #`EdgesToBeAdded` **then**
    `EdgesToBeAdded` = non-forbidden edges in G^c incident to w
for all triangle obstructions (v, uvw) **do**
  for all good (v, uvw)-edges e **do**
    Add bad edges e to `NewForbiddenEdges`
  if #non-forbidden good (v, uvw)-edges < #`EdgesToBeAdded` **then**
    `EdgesToBeAdded` = non-forbidden good (v, uwv)-edges
for all 4-cycle obstructions $(u, uvwx)$ **do**
  for all good $(u, uvwx)$-edges e **do**
    Add bad edges e to `NewForbiddenEdges`
  if #non-forbidden good $(u, uvwx)$-edges < #`EdgesToBeAdded` **then**
    `EdgesToBeAdded` = non-forbidden good $(u, uwvx)$-edges
if we found any of the above obstructions **then**
  for all edges e in `EdgesToBeAdded` **do**
    AddEdges($G + e$, `NewForbiddenEdges`)
  return
else
  if G is K_2-hypohamiltonian **then**
    Output G
  for all edges e in `EdgesToBeAdded` **do**
    if e is not forbidden and $G + e$ is not hamiltonian **then**
      AddEdges($G + e$, `NewForbiddenEdges`)

cannot be hamiltonian. Hence, we add e on line 39, since we do this for all non-forbidden and non-bad edges in G'^c.

Now suppose that G' does contain an obstruction. Assume the algorithm found a type A obstruction (W, X) which has the fewest non-forbidden good edges among all obstructions of the graph that were checked. By Lemma 3.2, (W, X) is not a type A obstruction in G. Hence, G contains a good (W, X) A-edge $e \in E(G) \setminus E(G')$. Since $G' + e$ is not hamiltonian, e is not forbidden and it is added to G' on line 32 of Algorithm 2. An analogous reasoning holds for the other obstructions.

Thus, we call Algorithm 2 on a graph isomorphic to a spanning subgraph of G with $m + 1$ edges. By induction, we will call Algorithm 2 on a graph isomorphic to G, say H, and since it is K_2-hypohamiltonian, we will output H on line 35. □

3.2.3 Results

We created an implementation of this algorithm which can be found on GitHub [58] and used this to improve the results given in [53]. We generated all pairwise non-isomorphic K_2-hypohamiltonian graphs of order n for all $n \leq 19$. Note that previously this was only done up to order 13. We were also able to extend the results for given lower bounds on the girth. Indeed, since we only add edges in the algorithm, we can simply prune the search as soon as a cycle of forbidden length appears. The longest computations were parallellised and performed on the supercomputer of the VSC (Flemish Supercomputer Center) as they took 1-2 CPU-years. The results are summarised in Table 3.1. The complete lists can be downloaded from the *House of Graphs* [30] at `https://houseofgraphs.org/meta-directory/K2-hypohamiltonian`.

In particular, we show that there exist no K_2-hypohamiltonian graphs of order 14 or order 17. Combined with our results from [53], we can now characterise the orders for which there exist K_2-hypohamiltonian graphs, thereby mirroring the characterisation of orders of hypohamiltonian graphs given by Aldred, McKay and Wormald [2].

Theorem 3.12. *There exists a K_2-hypohamiltonian graph of order n if and only if $n \in \{10, 13, 15, 16\}$ or $n \geq 18$.*

Currently it is unknown whether K_2-hypohamiltonian graphs of girth 3 or girth 4 exist. This is different from the hypohamiltonian setting where the smallest hypohamiltonian graph of girth 3 and the smallest of girth 4 have order 18. From Table 3.1 we infer the following.

Order	$g \geq 3$	$g \geq 4$	$g \geq 5$	$g \geq 6$	$g \geq 7$
10	1	1	1	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	1	1	1	0	0
14	**0**	0	0	0	0
15	**1**	1	1	0	0
16	**4**	4	4	0	0
17	**0**	**0**	0	0	0
18	**3**	**3**	3	0	0
19	**28**	**28**	28	0	0
20	?	**2**	2	0	0
21	?	?	31	0	0
22	?	?	**332**	0	0
23	?	?	**19**	0	0
24	?	?	**613**	0	0
25	?	?	?	**1**	0
26	?	?	?	**0**	**0**
27	?	?	?	**0**	**0**
28–30	?	?	?	?	**0**

Table 3.1: Counts of K_2-hypohamiltonian graphs. Columns with $g \geq k$ indicate that these are only the graphs with girth at least k. Bold entries indicate new results.

Proposition 3.13. *A K_2-hypohamiltonian graph of girth* 3 *has order at least* 20*. A K_2-hypohamiltonian graph of girth* 4 *has order at least* 21*.*

We remarked in [53] that the smallest K_2-hypohamiltonian graph which is not hypohamiltonian has order at least 14 and at most 18. This upper bound comes from a graph of girth 5. As our generation results did not yield any K_2-hypohamiltonian graphs of girth 3 or 4 up to order 18, we have shown the following.

Proposition 3.14. *The smallest non-hypohamiltonian K_2-hypohamiltonian graph has order* 18*.*

There is only one such graph of order 18 and it is obtained by applying the operation detailed in Section 3.3 to two copies of the Petersen graph, see [53, Figure 4b][2].

[2]This graph can also be inspected on the *House of Graphs* [30] at `https://houseofgraphs.org/graphs/49001`.

In [59] it is shown that a smallest planar hypohamiltonian graph has order at least 23, improving a result by Aldred, McKay and Wormald [2]. In [53] it was proven that a smallest planar K_2-hypohamiltonian graph has order at least 18 and at most 48. There is no mention of planar K_2-hypohamiltonian graphs of girth at least 4. We improve the lower bound on the order of a smallest planar K_2-hypohamiltonian graph using our generator and also give a lower bound when the girth is at least 4.

Note that it is relatively simple and efficient to restrict the algorithm to the class of planar graphs. In particular, checking planarity at the start of Algorithm 2 and pruning if the graph is non-planar will generate all planar K_2-hypohamiltonian graphs. We implemented this using the planarity algorithm distributed with `nauty` [100] based on a method by Boyer and Myrvold [15] and this allowed us to show the following.

Proposition 3.15. *A smallest planar K_2-hypohamiltonian graph has order at least* 24*. A smallest planar K_2-hypohamiltonian graphs of girth at least* 4 *has order at least* 26*.*

The smallest planar K_2-hypohamiltonian graph of girth 5 was determined by the authors in [53]. It has 48 vertices[3]. Note that 3-connected planar graphs have girth at most 5, hence there are no planar K_2-hypohamiltonian graphs of higher girth.

Similarly, we can extend our algorithm to exhaustively search for all bipartite K_2-hypohamiltonian graphs. Since we start from a cycle and a K_2 in the algorithm there are two options. If the order is odd, the cycle will be odd and the graph can never become bipartite. If the order is even, the graph is bipartite and after adding the first edge with endpoint on the cycle and on the K_2 the colour classes are completely fixed. We can then add all edges between vertices of the same colour class to the list of forbidden edges. Our generator will then only add edges that preserve bipartiteness, which allows to advance several orders further than in the general case.

For a bipartite K_2-hypohamiltonian graph G it must hold that G is *balanced*, i.e. both colour classes have the same number of vertices, hence that every vertex-deleted subgraph is non-hamiltonian. A hypohamiltonian graph cannot be bipartite, since we cannot have for all vertices that their removal yields a balanced bipartite graph. However, a question by Grötschel [68, Problem 4.56] in the same vein asked whether there exist bipartite hypotraceable graphs, i.e. a graph for which every vertex-deleted subgraph has a hamiltonian path, but which itself does not contain one.

[3]This graph can also be inspected on the *House of Graphs* [30] at `https://houseofgraphs.org/graphs/49121`.

Our implementation yields the following results.

Proposition 3.16. *A smallest bipartite K_2-hypohamiltonian graph has order at least* 30*. If its girth is at least* 6 *its order is at least* 32*. If its girth is at least* 8 *its order is at least* 36*.*

The correctness of our generation algorithm was proven in Theorem 3.11. To verify we did not make any implementation mistakes, we performed various correctness tests.

In [53] we already gave counts for K_2-hypohamiltonian graphs for given lower bounds on the girth. Our results coincide.

It is important to note that we use the same routines for checking hamiltonicity and K_2-hypohamiltonicity as in that paper. As mentioned in [53], these were extensively tested using independent programs.

It is easy to adapt our algorithm to only generate graphs with minimum degree at least x and maximum degree at most y. Using this we can generate cubic K_2-hypohamiltonian graphs. Since there exist very efficient generators for cubic graphs (e.g. `snarkhunter` [19]), using one of these and then filtering the K_2-hypohamiltonian graphs is much faster. Counts of cubic K_2-hypohamiltonian graphs were given in [53]. We adapted our algorithm and generated all cubic K_2-hypohamiltonian graphs up to order 22. The results are the same in both cases.

Our implementation of the algorithm is open source software and can be found on GitHub [58] where it can be verified and used by other researchers.

3.3 Infinite families of K_2-hypohamiltonian graphs

3.3.1 Amalgam of K_2-hypohamiltonian graphs

Next to the exhaustive generation of K_2-hypohamiltonian graphs, operations preserving K_2-hypohamiltonicity are of interest in order to describe infinite families. Here, we introduce an operation creating graphs from smaller ones which will preserve K_2-hypohamiltonicity as well as planarity under certain conditions. Using this we improve a result from [53] attempting to characterise the orders for which planar K_2-hypohamiltonian graphs exist.

Let G be a graph and $a, a', b, b' \in V(G)$ be pairwise distinct vertices. We say the tuple (G, a, a', b, b') *satisfies the gluing property* if

- a and b have degree 3 and a' and b' have degree at least 3,
- $aa', bb', ab \in E(G)$ and $ab', a'b, a'b' \notin E(G)$.

Let $(G_i, a_i, a_i', b_i, b_i')$, for $i = 1, 2$, be two tuples satisfying the gluing property and $E_1 = E(G_1) \setminus \{a_1b_1, b_1b_1'\}$ and $E_2 = E(G_2) \setminus \{a_2b_2, b_2b_2'\}$. Take the disconnected graph

$$(V(G_1) \cup V(G_2), E_1 \cup E_2 \cup \{b_1b_2, b_1b_2', b_1'b_2\})$$

and identify a_1 with a_2 and a_1' with a_2' to get a connected graph G. We say G is the *amalgam* of G_1 and G_2 if it is clear which tuples we started with and we define the vertices $a := a_1 = a_2$, $a' := a_1' = a_2'$ of G.

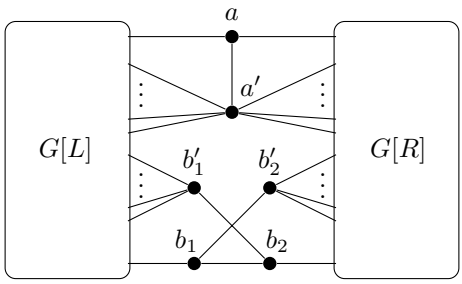


Figure 3.1: Visualisation of the amalgam G of G_1 and G_2.

In what follows, we show that under certain constraints this operation preserves non-hamiltonicity, K_2-hamiltonicity and planarity. To aid in the notation of the proofs, we define subsets of $V(G)$:

$$L := V(G_1) \setminus \{a_1, a_1', b_1, b_1'\}, \quad R := V(G_2) \setminus \{a_2, a_2', b_2, b_2'\}.$$

Let S be a subset of $V(G) - L - R$ and $\mathfrak{h}$ be a cycle in G. Suppose we have a pair of edges (lx, ry) such that $l \in L$, $r \in R$ and $x, y \in S$ such that lx and ry are edges in $\mathfrak{h}$ and $x = y$ or there is an xy-path in $\mathfrak{h}$ containing only vertices in S. Then we call this pair a *traversal* of $\mathfrak{h}$ through S.

Let $s_1, \ldots, s_4$ be all pairwise different traversals of some cycle $\mathfrak{h}$ through $\{a, a'\}$ up to symmetry, i.e. up to switching L and R, defined in Figure 3.2. Let $t_1, \ldots, t_4$ be all pairwise different traversals of some cycle $\mathfrak{h}$ through $\{b_1, b_1', b_2, b_2'\}$ up to symmetry defined in Figure 3.2. For s_i, $i = 1, \ldots, 4$, we define the *traversal number* $\phi(s_i)$ to be the number of traversals of $\mathfrak{h}$ through $\{a, a'\}$, e.g. $\phi(s_1) = 2$, $\phi(s_3) = 0$. Similarly for t_i, $i = 1, \ldots, 4$ we define $\phi(t_i)$ the be the number of traversals of $\mathfrak{h}$ through $\{b_1, b_1', b_2, b_2'\}$, e.g. $\phi(t_1) = 1$.

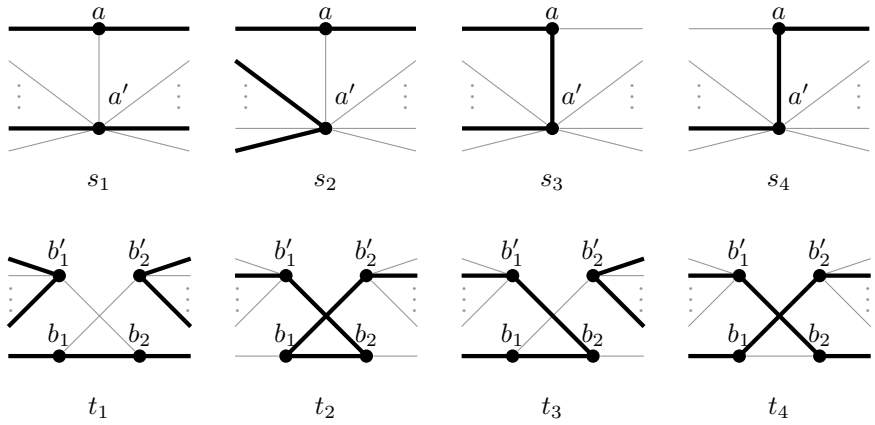


Figure 3.2: All pairwise different traversals of some cycle $\mathfrak{h}$ (in black) through $\{a, a'\}$ (top row) and through $\{b_1, b_1', b_2, b_2'\}$ (bottom row) up to symmetry. Edges to the left are going to some vertex in L, edges to the right are going to some vertex in R.

Lemma 3.17. *Let $(G_i, a_i, a_i', b_i, b_i')$, for $i = 1, 2$, be two tuples satisfying the gluing property such that G_1 and G_2 are non-hamiltonian. Then the amalgam G of G_1 and G_2 is non-hamiltonian.*

Proof. Suppose that G does contain a hamiltonian cycle $\mathfrak{h}$. It must pass from L to R and back (at least once) via precisely one s_i and one t_i configuration of Figure 3.2. For $i, j = 1, \ldots, 4$, we have that $\phi(s_i) + \phi(t_j)$ must be even and at least 2 for all such i and j. It remains to show all possible combinations up to symmetry lead to a contradiction.

Suppose we are in the case of s_1 and t_3. Then $\mathfrak{h}$ consists of two aa'-paths, one of which contains a, a', all vertices of L and b_1', b_2 and b_1. Removing b_2 and adding edges aa' and b_1b_1' to this path yields a hamiltonian cycle in G_1, a contradiction. In the case of s_1 and t_4, we can split $\mathfrak{h}$ into four paths, two of which are a k_1l_1-path and a k_2l_2-path where all interior vertices are in L and $\{k_1, k_2\} = \{a, a'\}$ and $\{l_1, l_2\} = \{b_1, b_1'\}$. Adding aa' and b_1b_1' to the union of these two paths yields a hamiltonian cycle in G_1.

In the case of s_2 and t_1, we get an ab_1-subpath of $\mathfrak{h}$ where the interior vertices are vertices of L or a' or b_1'. Adding ab_1 to this path yields a hamiltonian cycle in G_1. In the case of s_2 and t_2 we get an ab_1-subpath of $\mathfrak{h}$ in which the interior vertices are vertices of L, a', b_1' or b_2. Removing b_2 and adding edges b_1b_1' and ab_1 to this path yields a hamiltonian cycle in G_1.

In the case of s_3 and t_4, we get a b_1b_1'-subpath of $\mathfrak{h}$ in which the interior vertices are a, a' and vertices of L. Adding edge b_1b_1' to this path yields a hamiltonian cycle in G_1.

In the case of s_4 and t_1, we get an ab_1-subpath of $\mathfrak{h}$ in which the interior vertices are a', b_1' and vertices of L. Adding edge ab_1 to this path gives us a hamiltonian cycle in G_1. In the case of s_4 and t_2, we get an ab_1-subpath of $\mathfrak{h}$ in which the interior vertices are a', b_1', b_2 or vertices of L. Removing b_2 and adding edges ab_1 and b_1b_1' to this path yields a hamiltonian cycle in G_1.

Since the unmentioned cases have either $\phi(s_i) + \phi(t_j)$ odd or zero, we conclude that G must be non-hamiltonian. □

In the following proofs, we will often create a cycle C in G from cycles or paths in G_1 and G_2. To make this more concise we define the following notation. Let H_1 be a subgraph of G_1; H_2 be a subgraph of G_2; and $e_1, \dots, e_m$ be edges in G. We can identify $H_1 - b_1b_1' - a_1b_1$ with a subgraph of G. Similarly, we can identify $H_2 - b_2b_2' - a_2b_2$ with a subgraph in G. We define $C(H_1, H_2, e_1, \dots, e_m)$ to be the subgraph in G, which is the union of both of these identified subgraphs in G together with the edges $e_1, \dots, e_m$. We allow this set of edges to be empty in which case we write $C(H_1, H_2)$.

Lemma 3.18. *Let $(G_i, a_i, a_i', b_i, b_i')$, for $i = 1, 2$, be two tuples satisfying the gluing property such that G_1 and G_2 are K_2-hamiltonian, $G_i - a_i'$ has at least two hamiltonian cycles, one containing the edge b_ib_i' and one which does not, and $G_i - v$ is hamiltonian for all $v \in N_{G_i}[b_i]$. Then the amalgam G of G_1 and G_2 is K_2-hamiltonian.*

Proof. We show for all adjacent $u, v \in V(G)$ that $G - u - v$ is hamiltonian.

Suppose $u, v \notin \{a, a', b_i, b_i' \mid i = 1, 2\}$. Without loss of generality assume that $u, v \in L$. There exists a hamiltonian cycle $\mathfrak{h}_1$ in $G_1 - u - v$. Suppose $a_1b_1 \notin E(\mathfrak{h}_1)$. Then $\mathfrak{h}_1$ must contain b_1b_1' as b_1 has degree 3. We also have a hamiltonian cycle $\mathfrak{h}_2$ in $G_2 - a_2 - a_2'$, which must contain b_2b_2'. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2', b_1'b_2)$ is a hamiltonian cycle in $G - u - v$. Conversely, if $\mathfrak{h}_1$ does contain a_1b_1, it can either contain b_1b_1' or not. Suppose the former, then let $\mathfrak{h}_2$ be a hamiltonian cycle containing b_2b_2' in $G_2 - a_2'$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2, b_1b_2', b_1'b_2)$ is a hamiltonian cycle in $G - u - v$. Finally, suppose that $\mathfrak{h}_1$ contains a_1b_1 but not b_1b_1'. Let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a_2'$ not containing b_2b_2', then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2)$ is a hamiltonian cycle in $G - u - v$.

Suppose $u = a$ and $v \in N_G(a)$. Without loss of generality assume that $v \in L \cup \{a'\}$. Let $\mathfrak{h}_1, \mathfrak{h}_2$ be hamiltonian cycles in $G_1 - a_1 - v$ and $G_2 - a_2 - a_2'$, respectively. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2', b_1'b_2)$ is a hamiltonian cycle in $G - u - v$.

Suppose $u \neq a$, $v = a'$. Without loss of generality assume that $u \in L$. Let $\mathfrak{h}_1$ be a hamiltonian cycle in $G_1 - u - a'_1$. It can either contain $b_1b'_1$ or not. In the former case, let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a'_2$ containing $b_2b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2, b_1b'_2, b'_1b_2)$ is a hamiltonian cycle in $G - u - v$. In the latter case, let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a'_2$ not containing $b_2b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2)$ is a hamiltonian cycle in $G - u - v$.

Suppose $u = b_1$ and $v = b_2$. Let $\mathfrak{h}_1, \mathfrak{h}_2$ be hamiltonian cycles in $G_1 - b_1$ and $G_2 - b_2$, respectively. Then $C(\mathfrak{h}_1, \mathfrak{h}_2) - aa'$ is a hamiltonian cycle in $G - u - v$.

Suppose $u = b_1$ and $v = b'_2$. Let $\mathfrak{h}_1$ be a hamiltonian cycle in $G_1 - a'_1$ containing $b_1b'_1$ and let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b'_1b_2) - b_1$ is a hamiltonian cycle in $G - u - v$.

Suppose $u = b_1$ and $v \notin \{b_2, b'_2\}$. Then $v \in L$. Let $\mathfrak{h}_1$ be a hamiltonian cycle in $G_1 - v$ and let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a'_2$ not containing $b_2b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b'_1b_2) - b_1$ is a hamiltonian cycle in $G - u - v$.

Finally, let $u \neq b_2$ and $v = b'_1$. Then $u \in L$. Let $\mathfrak{h}_1$ be a hamiltonian cycle in $G_1 - u - v$ and $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a'_2$ not containing $b_2b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2)$ is a hamiltonian cycle in $G - u - v$.

Since all unmentioned cases are symmetric to the ones above, we conclude that G is K_2-hamiltonian. □

Lemma 3.19. *Let $(G_i, a_i, a'_i, b_i, b'_i)$, for $i = 1, 2$, be two tuples satisfying the conditions of Lemma 3.17 and G be the amalgam of G_1 and G_2. For $v \in V(G_i - b_i)$, if $G_i - v$ is hamiltonian, so is $G - v$. Moreover, for any edge $e \in E(G_i - b_i)$ which is also an edge of G, if there is a hamiltonian cycle in $G_i - v$ containing e (not containing e), then there exists a hamiltonian cycle in $G - v$ containing e (not containing e). Lastly, $G - b_i$ is non-hamiltonian.*

Proof. Without loss of generality, let $i = 1$. Let $v \in V(G_1 - b_1)$ and suppose $\mathfrak{h}_1$ is a hamiltonian cycle in $G_1 - v$.

Assume $a_1b_1 \notin E(\mathfrak{h}_1)$. Let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a_2 - a'_2$. We see that $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b'_2, b'_1b_2)$ is a hamiltonian cycle in $G - v$. Conversely, suppose that $a_1b_1 \in E(\mathfrak{h}_1)$. Then either $b_1b'_1 \in E(\mathfrak{h}_1)$ or not. Suppose the former and let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a'_2$ containing $b_2b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2, b_1b'_2, b'_1b_2)$ is a hamiltonian cycle in $G - v$. Suppose the latter and let $\mathfrak{h}_2$ be a hamiltonian cycle in $G_2 - a'_2$ not containing $b_2b'_2$. Then $C(\mathfrak{h}_1, \mathfrak{h}_2, b_1b_2)$ is a hamiltonian cycle in $G - v$.

Note that by construction of these cycles any edge of G_1 on $\mathfrak{h}_1$ which is an edge in G lies on the hamiltonian cycle of $G - v$ and any edge of G_1 not on $\mathfrak{h}_1$ does not lie on the hamiltonian cycle in $G - v$.

Now suppose there exists a hamiltonian cycle $\mathfrak{h}$ in $G - b_1$. Then $b_1'b_2 \in E(\mathfrak{h})$ and $ub_2'w$ is a subpath of $\mathfrak{h}$, where u and w are vertices of R. Denote the traversals of $\mathfrak{h}$ through $\{b_1', b_2, b_2'\}$ by t_5. Then $\phi(t_5) = 1$. We show that all a priori possible traversals of $\mathfrak{h}$ through $\{a, a'\}$ lead to a contradiction. Since $\phi(s_i) + \phi(t_5)$ is even and at least 2, we only need to look at s_2, s_4 and their mirror images, i.e. with L and R swapped.

Assume we are in case s_2 or s_4 and t_5, then the intersection of $\mathfrak{h}$ with G_1 yields an a_1b_1'-path in which the interior vertices are precisely a_1' and the vertices of L. Adding edges b_1b_1' and a_1b_1 to this path yields a hamiltonian cycle in G_1, a contradiction.

Suppose we are in the case where s_2 is mirrored or s_4 is mirrored and t_5. Then the intersection of $\mathfrak{h}$ with G_2 yields an a_2b_2-path in which the interior vertices are precisely a_2', b_2' and all vertices of R. Adding edge a_2b_2 to this path leads to a hamiltonian cycle in G_2, a contradiction. □

This lemma shows that the operation, when applied to hypohamiltonian graphs, never yields a hypohamiltonian graph.

Let G be a planar graph and let V_F be the set of all vertices that lie on the boundary of a face F in a plane embedding of G. We say vertices $v_0, \ldots, v_{k-1} \in V(G)$ are *co-facial* in a plane embedding of G if they all belong to the same set V_F for some face F of the embedding. We say $v_0, \ldots, v_{k-1}$ *appear in order* in F if for every $i = 0, \ldots, k-1$ there is a v_iv_{i+1}-path, taking indices mod k, containing only edges on the boundary of F such that this path does not contain the vertex v_j for any $j = 0, \ldots, i-1, i+2, \ldots, k-1$.

Lemma 3.20. *Let $(G_i, a_i, a_i', b_i, b_i')$, for $i = 1, 2$, be two tuples satisfying the gluing property such that G_1 and G_2 are planar. If G_1 admits a plane embedding such that a_1', a_1, b_1, b_1' are co-facial and appear in this order and G_2 admits a plane embedding such that a_2, a_2', b_2, b_2' are not co-facial, then the amalgam G of G_1 and G_2 is a planar graph.*

Proof. As a_2 and b_2 have degree 3 in G_2, a_2', a_2, b_2 and a_2, b_2, b_2' are co-facial in G_2 and appear in these orders in their respective faces. Take a plane embedding of G_1 for which the vertices a_1, a_1', b_1, b_1' lie on the outer face. Note that this is always possible. Hence, since G_1 and G_2 are plane, removing a_ib_i and b_ib_i' from G_i for $i = 1, 2$ yields two plane graphs such that a_i, a_i', b_i, b_i' are co-facial for $i = 1, 2$. This is depicted at the top of Figure 3.3. Denote these new graphs by

G_1' and G_2', respectively. Identifying a_1 with a_2 and a_1' with a_2' in the disjoint union of G_1' and G_2' also yields a plane graph in which the following vertices are co-facial and appear in order $a', b_1', b_1, a, b_2', b_2$ as can be seen at the bottom of Figure 3.3. Hence, we can add edges b_1b_2, b_1b_2' and $b_1'b_2$ without creating any crossings. □

In [53], the authors showed the existence of a planar K_2-hypohamiltonian graph for every order from 177 onwards. Using the operation described above, we can improve this bound.

Consider a graph G containing a 5-cycle $C = v_0 \dots v_4$ such that every v_i is cubic. Denote the neighbour of v_i not on C by v_i'. Then the cycle C is called *extendable* if for any i, taking indices mod 5, (i) there exists a hamiltonian cycle $\mathfrak{h}$ in $G - v_i$ with $v_{i-2}v_{i+2} \notin E(\mathfrak{h})$ and (ii) there exists a hamiltonian cycle $\mathfrak{h}'$ in $G - v_i'$ with $\mathfrak{h}' \cap C = v_{i-2}v_{i-1}v_iv_{i+1}v_{i+2}$. From Lemma 3.19 we immediately get the following.

Corollary 3.21. *Let $(G_i, a_i, a_i', b_i, b_i')$, for $i = 1, 2$, be two tuples satisfying the conditions of Lemma 3.19 such that G_1 contains an extendable 5-cycle $v_0 \dots v_4$. If a_1, a_1', b_1 and b_1' are disjoint from $v_0, \dots, v_4$ and their neighbours, then the amalgam G of G_1 and G_2 contains an extendable 5-cycle.*

Theorem 3.22. *There exists a planar K_2-hypohamiltonian graph of order n if $n \geq 134$.*

Proof. In [53] it was shown that this holds for $n \geq 177$. Starting from the five planar K_2-hypohamiltonian graphs of Figure 6 in that paper and gluing them together in the appropriate way using Lemma 4 from that paper yields planar K_2-hypohamiltonian graphs containing an extendable 5-cycle on orders 134–149. We will show the existence of planar K_2-hypohamiltonian graphs containing an extendable 5-cycle on orders 150–153. Denote by G_{50}, G_{52} and G_{53}, respectively, the graphs of Figure 3.4. In Appendix B.1, we show that these three planar K_2-hypohamiltonian graphs have a tuple $(G_i, a_i, a_i', b_i, b_i')$ for $i = 50, 52, 53$, that satisfies the gluing property and in which $G_i - a_i'$ has at least two hamiltonian cycles, one containing the edge b_ib_i' and one which does not and in which $G_i - v$ is hamiltonian for all $v \in N_{G_i}[b_i]$. Moreover, $(G_{52}, c_{52}, c_{52}', d_{52}, d_{52}')$ is a tuple satisfying the same properties, such that $a_{52}, a_{52}', b_{52}, b_{52}'$ and $c_{52}, c_{52}', d_{52}, d_{52}'$ are pairwise disjoint. We see that there is an embedding such that a_i, a_i', b_i, b_i' are co-facial for $i = 50, 52$ and $c_{52}, c_{52}', d_{52}, d_{52}'$ and $a_{53}, a_{53}', b_{53}, b_{53}'$ are not co-facial.

The amalgam G_{100} of $(G_{50}, a_{50}, a_{50}', b_{50}, b_{50}')$ and $(G_{52}, c_{52}, c_{52}', d_{52}, d_{52}')$ is by Lemma 3.17 and Lemma 3.18 a K_2-hypohamiltonian graph. By Lemma 3.20 it

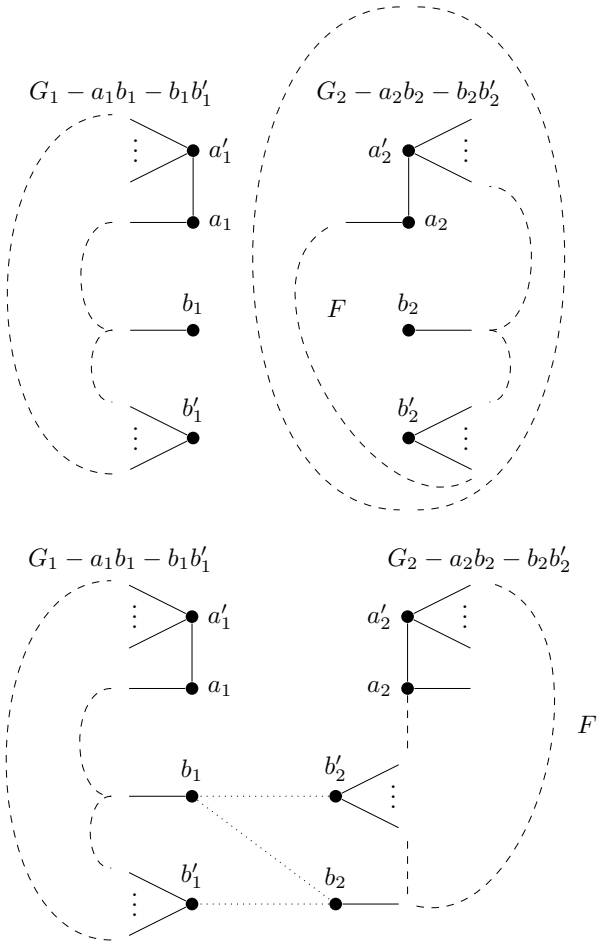


Figure 3.3: A visualisation of the amalgam of G_1 and G_2. Dashed lines represent (part of) the boundary of some face in the embedding. The top image depicts G_1, with a_1', a_1, b_1, b_1' lying on the boundary of the outer face, and G_2 after the removal of a_ib_i and b_ib_i'. We see that a_1', a_1, b_1, b_1' still lie on the outer face and that a_2', a_2, b_2, b_2' now lie on the boundary of some face F. In the bottom image we have taken the embedding of $G_2 - a_2b_2 - b_2b_2'$ for which F is the outer face. It is now easily seen that the identification of a_1' with a_2' and a_1 with a_2 leaves a plane graph and that the addition of the dotted edges $b_1b_2, b_1'b_2$ and b_1b_2' does as well.

is planar and by Corollary 3.21 it contains an extendable 5-cycle. Moreover, by Lemma 3.19 $(G_{100}, a_{52}, a_{52}', b_{52}, b_{52}')$ satisfies all the conditions of Lemma 3.18.

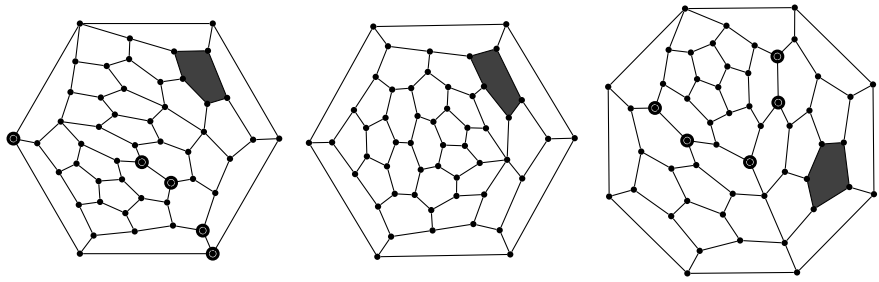


Figure 3.4: Planar K_2-hypohamiltonian graphs on 50, 52, and 53 vertices. A face for which the boundary is an extendable 5-cycle is filled in. All vertices for which the vertex-deleted subgraphs are non-hamiltonian are circled. Proofs of these facts are given in [53].

It is easy to see that this extendable 5-cycle and $a_{52}, a'_{52}, b_{52}, b'_{52}$ are disjoint and that the latter vertices are co-facial. Hence, the amalgam of $(G_{100}, a_{52}, a'_{52}, b_{52}, b'_{52})$ and $(G_{52}, c_{52}, c'_{52}, d_{52}, d'_{52})$ is then a planar K_2-hypohamiltonian graph of order 150 containing an extendable 5-cycle by the same reasons as before.

Similarly, we get a planar K_2-hypohamiltonian graph of order 151 with an extendable 5-cycle by taking the amalgam of $(G_{100}, a_{52}, a'_{52}, b_{52}, b'_{52})$ and $(G_{53}, a_{53}, a'_{53}, b_{53}, b'_{53})$.

We get planar K_2-hypohamiltonian graphs of order 152 and 153 containing an extendable 5-cycle by taking the amalgam of the appropriate tuples of G_{52} with itself and then taking the amalgam with either G_{52} or G_{53}. Applying, ad infinitum, the dodecahedron operation described in [143] to these twenty planar K_2-hypohamiltonian graphs, each with an extendable 5-cycle and having consecutive orders, yields the result. □

3.3.2 K_2-hypohamiltonian graphs with large maximum degree

Thomassen showed in [131] that the maximum degree and maximum number of edges in an n-vertex hypohamiltonian graph can be $n/2-9$ and $(n-20)^2/4+32$. A similar result is not known for K_2-hypohamiltonian graphs, since no non-trivial upper bound on the maximum degree and size of such a graph is known. The third author has shown in [143] that for any integer $d \geq 3$ there exists a K_2-hypohamiltonian graph with maximum degree d. But in that construction the order of the graph becomes very large as d increases. In this section, we show

the existence of an infinite family consisting of n-vertex K_2-hypohamiltonian graphs with maximum degree $(n-1)/3$ and maximum size $2n-5$. This family contains both the smallest (the Petersen graph) and second smallest (K_2-)hypohamiltonian graphs and thus can be seen as yet another generalisation of Petersen's famous graph.

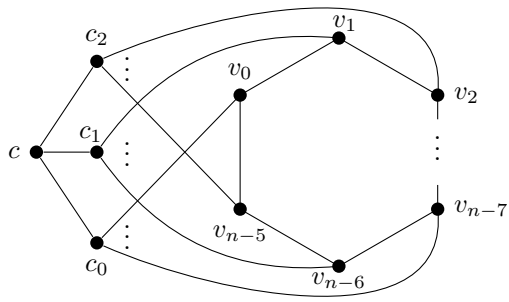


Figure 3.5: Visualisation of a graph with order n from Theorem 3.23. A vertex v_i is adjacent to c_j if $i \equiv j \bmod 3$.

Theorem 3.23. *For every integer $n \geq 10$ which is congruent to 1 mod 3, there exists a hypohamiltonian K_2-hamiltonian graph of order n, size $2n-5$ and maximum degree $(n-1)/3$.*

Proof. Consider the vertex set

$$V_n := \{c, c_0, c_1, c_2\} \cup \{v_i\}_{i=0}^{n-5}$$

and the edge set

$$E_n := \{c_0c, c_1c, c_2c, v_0v_{n-5}\} \cup \{v_iv_{i+1}\}_{i=0}^{n-6} \cup \{v_ic_{i \bmod 3}\}_{i=0}^{n-5}.$$

We define the family

$$\mathcal{F} := \{(V_n, E_n) \mid n = 3m+10, m \geq 0\}.$$

Every graph G in $\mathcal{F}$ is non-hamiltonian. Indeed, suppose that G contains a hamiltonian cycle $\mathfrak{h}$ and let the vertices of G be labelled as above. In the remainder of this proof, we will assume all indices are taken modulo $n-4$. Without loss of generality assume that $\mathfrak{h}$ contains c_1cc_2 and $v_ic_0v_j$. Then $i \equiv j \equiv 0 \bmod 3$ and $\mathfrak{h}$ must contain either $v_iv_{i+1}\dots v_{j-1}$ or $v_iv_{i-1}\dots v_{j+1}$. Indeed, suppose $\mathfrak{h}$ contains $v_iv_{i+1}\dots v_{j-k}$, but not $v_{j-k}v_{j-k+1}$, for $k>1$, then $\mathfrak{h}$ can only contain either the vertices $v_{j-k+1},\dots,v_{j-1}$ or the vertices $v_{j+1},\dots,v_{i-1}$. In both cases $\mathfrak{h}$ is not hamiltonian. Similarly, we get a contradiction if $\mathfrak{h}$ contains

$v_iv_{i-1}\dots v_{j+k}$, but not $v_{j+k}v_{j+k-1}$. Now suppose $\mathfrak{h}$ does contain $v_iv_{i+1}\dots v_{j-1}$, then $\mathfrak{h}$ must also contain $v_jv_{j+1}\dots v_{i-1}$. However, $i-1 \equiv j-1 \equiv 2 \bmod 3$, which means there is no pair of edges $v_{i-1}c_1$, $v_{j-1}c_2$ or $v_{i-1}c_2$, $v_{j-1}c_1$ for which both edges are present in G, a contradiction. In the case where $\mathfrak{h}$ contains $v_iv_{i-1}\dots v_{j+1}$, we have the same situation.

We now show that any vertex-deleted subgraph contains a hamiltonian cycle. Denote the order of G by n. Suppose $v = c$, then $G - v$ has hamiltonian cycle

$$v_0c_0v_3v_4c_1v_1v_2c_2v_5\dots v_{n-5}.$$

Suppose $v = c_2$, then $G - v$ has the hamiltonian cycle

$$v_0c_0cc_1v_1\dots v_{n-5}.$$

For the vertices c_0 and c_1, the arguments are analogous. Suppose $v = v_i$ and assume without loss of generality that $i \equiv 0 \bmod 3$, then $G - v$ has the hamiltonian cycle

$$v_{i-1}c_2v_{i+2}v_{i+1}c_1cc_0v_{i+3}v_{i+4}\dots v_{i-2}.$$

The remaining cases are analogous, hence, $G-v$ is hamiltonian for all $v \in V(G)$.

Similarly, we show G is K_2-hamiltonian. Suppose $v = c, w = c_2$. Then $G-v-w$ has the hamiltonian cycle

$$v_0v_1c_1v_4v_3v_2c_2v_5\dots v_{n-5}.$$

Suppose $v = c_1$, $w = v_i$, where $i \equiv 1 \bmod 3$. Then $G-v-w$ has the hamiltonian cycle

$$v_{i-1}c_0cc_2v_{i+1}v_{i+2}\dots v_{i-2}.$$

Suppose $v = v_i$, $w = v_{i+1}$, such that $i \equiv 0 \bmod 3$. Then $G - v - w$ has a hamiltonian cycle

$$v_{i-1}c_2v_{i+2}v_{i+3}c_0cc_1v_{i+4}v_{i+5}\dots v_{i-2}.$$

All other cases are analogous to one of these, hence G is K_2-hamiltonian. Hence, G is a hypohamiltonian K_2-hamiltonian graph.

Now let $n \geq 10$ and $n \equiv 1 \bmod 3$, i.e. $n = 3m + 10$ for some integer $m \geq 0$. Then (V_n, E_n) has maximum degree $m + 3$. □

Experiments indicate that for the same orders there also exists an infinite family of hypohamiltonian K_2-hypohamiltonian graphs in which a member of order n has size $2n - 4$, thus slightly beating the bound above. However, the proofs of e.g. non-hamiltonicity are far more tedious, hence we have decided to omit this.

Acknowledgements

Several of the computations for this work were carried out using the supercomputer infrastructure provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government. The research of Jan Goedgebeur and Jarne Renders was supported by Internal Funds of KU Leuven. The research of Carol T. Zamfirescu was supported by a Postdoctoral Fellowship of the Research Foundation - Flanders (FWO).

Part II

2-Factors in Graphs

Chapter 4

The Frank number and nowhere-zero flows on graphs

This chapter is a minimally edited version of the following published work[1].

J. Goedgebeur, E. Máčajová and J. Renders. The Frank number and nowhere-zero flows on graphs. *Eur. J. Comb.*, 126:104127, 2025. DOI: 10.1016/j.ejc.2025.104127

My contributions to this paper, in accordance with the contributor role taxonomy (CRediT)[2], consist of: Conceptualisation, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review and Editing, and Visualisation.

4.1 Introduction

An *orientation* (G, o) of a graph G is a directed graph with vertices $V(G)$ such that each edge $uv \in E(G)$ is oriented either from u to v or from v to u by the function o. An orientation is called *strong* if, for every pair of distinct vertices u and v, there exists an oriented uv-path, i.e. an oriented path starting at vertex

[1] An extended abstract of this work also appeared in the proceedings of the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2023) [44].

[2] See https://credit.niso.org/.

u and ending at vertex v. It is not difficult to see that an orientation is strong if and only if each edge cut contains edges oriented in both directions.

An edge e is *deletable* in an orientation (G, o) if the restriction of o to $E(G)-\{e\}$ yields a strong orientation of $G - e$. Note that this implies that o is a strong orientation. In this paper, a *cycle* is a connected 2-regular graph. A graph in which the removal of fewer than k edges cannot separate the graph into two components which both contain a cycle, is called *cyclically k-edge-connected*. The *cyclic edge connectivity* of a graph G is the largest k for which G is cyclically k-edge-connected.

Inspired by a problem of Frank, in 2021, Hörsch and Szigeti [79] proposed a new parameter for 3-edge-connected graphs called the Frank number. This parameter can be used to refine a theorem by Nash-Williams [101] stating that a graph has a k-arc-connected orientation if and only if it is $2k$-edge-connected.

Definition 4.1. *For a 3-edge-connected graph G, the* Frank number *– denoted by $fn(G)$ – is the minimum number k for which G admits k orientations such that every edge $e \in E(G)$ is deletable in at least one of them.*

Note that Definition 4.1 does not make sense for graphs which are not 3-edge-connected as such a graph G has at least one edge which is not deletable in any orientation of G.

A first general upper bound for the Frank number was established by Hörsch and Szigeti in [79]. They proved that $fn(G) \leq 7$ for every 3-edge-connected graph G. Moreover, in the same paper it is shown that the Berge–Fulkerson conjecture [120] implies that $fn(G) \leq 5$. We improve the former result by showing the following upper bound.

Theorem 4.2. *Every 3-edge-connected graph G has $fn(G) \leq 4$.*

We would also like to note that in [10] Barát and Blázsik very recently independently proved that $fn(G) \leq 5$ using methods similar to ours based on our Lemma 4.5 from this paper.

In [79], Hörsch and Szigeti also conjectured that every 3-edge-connected graph G has $fn(G) \leq 3$ and showed that the Petersen graph has Frank number equal to 3. In this paper we conjecture a stronger statement:

Conjecture 4.3. *The Petersen graph is the only cyclically 4-edge-connected graph with Frank number greater than 2.*

Barát and Blázsik showed in [11] that for any 3-edge-connected graph G, there exists a 3-edge-connected cubic graph H with $fn(H) \geq fn(G)$. Corollary 4.8

in Section 4.2.3 extends this result by showing that for any cyclically 4-edge-connected 3-edge-connected graph G, there exists a cyclically 4-edge-connected cubic graph H with $fn(H) \geq fn(G)$. Hence, it is enough to prove the conjecture for cubic graphs and in the remainder, we will mainly focus on them. Note that since cubic graphs cannot be 4-edge-connected, their Frank number is at least 2.

Hörsch and Szigeti proved in [79] that every 3-edge-connected 3-edge-colourable graph has Frank number at most 3. We remark that such graphs are always cubic. We strengthen this result by showing that these graphs have Frank number equal to 2. In fact, we prove the following more general theorem.

Theorem 4.4. *If G is a 3-edge-connected graph admitting a nowhere-zero 4-flow, then $fn(G) \leq 2$. In particular, $fn(G) = 2$ for every 3-edge-connected 3-edge-colourable graph G.*

It is also verified in [11] that several well-known infinite families of 3-edge-connected graphs have Frank number 2. This includes wheel graphs, Möbius ladders, prisms, flower snarks and an infinite subset of the generalised Petersen graphs. Note that except for the wheel graphs and flower snarks, these families all consist of 3-edge-colourable graphs. In the same paper it is also conjectured that every 3-edge-connected hamiltonian cubic graph has Frank number 2. Since every hamiltonian cubic graph is 3-edge-colourable, Theorem 4.4 also proves this conjecture.

The main tool in the proofs of the two mentioned results make use of nowhere-zero integer flows. We give a sufficient condition for an edge to be deletable in an orientation which is the underlying orientation of some all-positive nowhere-zero k-flow and construct two specific nowhere-zero 4-flows that show that the Frank number is 2.

Moreover, we also give two sufficient conditions for cyclically 4-edge-connected cubic graphs to have Frank number 2. We propose a heuristic algorithm and an exact algorithm for determining whether the Frank number of a 3-edge-connected cubic graph is 2. The heuristic algorithm makes use of the sufficient conditions mentioned earlier. Using our implementation of these algorithms we show that the Petersen graph is the only cyclically 4-edge-connected cubic graph up to 36 vertices with Frank number greater than 2. This implies a positive answer for Conjecture 4.3 up to this order in the family of cubic graphs.

After the introduction and preliminaries, our paper is divided into two main sections: Section 4.2 which is devoted to theoretical results and Section 4.3 which focuses on the algorithmic aspects of this problem. More precisely, in Section 4.2 we first prove our key Lemma 4.5 and use it to prove Theorems 4.2 and 4.4. Here, we also provide sufficient conditions for a cubic graph to have

Frank number 2. In Section 4.3 we describe the algorithms and use them to check Conjecture 4.3 for nontrivial non-3-edge-colourable cubic graphs up to 36 vertices. Together with our theoretical results this proves that there is no cubic counterexample to Conjecture 4.3 up to 36 vertices. We also note that an extended abstract of this paper appeared in the proceedings of WG 2023 [45].

4.1.1 Preliminaries

Let $\mathcal{H}$ be an abelian group. An *$\mathcal{H}$-flow* (o, f) on a graph G consists of an orientation (G, o) and a valuation $f : E(G) \to \mathcal{H}$ assigning elements of $\mathcal{H}$ to the edges of G in such a way that for every vertex v of G the sum of the values on the incoming edges is the same as the sum of the values on the outgoing edges from v. A $\mathbb{Z}$-flow is called a *k-flow* if the function f only takes values in $\{0, \pm 1, \pm 2 \ldots, \pm(k-1)\}$. An $\mathcal{H}$-flow (o, f) (or a k-flow) is said to be *nowhere-zero* if the value of f is not the identity element $0 \in \mathcal{H}$ ($0 \in \mathbb{Z}$) for any edge of $E(G)$.

A nowhere-zero k-flow on G is said to be *all-positive* if the value $f(e)$ is positive for every edge e of G. Every nowhere-zero k-flow can be transformed to an all-positive nowhere-zero k-flow by changing the orientation of the edges with negative $f(e)$ and changing negative values of $f(e)$ to $-f(e)$.

Let (G, o) be an orientation of a graph G. Let H be a subgraph of G. If the context is clear we write (H, o) to be the orientation of H where o is restricted to H. We define the set $D(G, o) \subseteq E(G)$ to be the set of all edges of G which are deletable in (G, o). Let $u, v \in V(G)$, if the edge uv is oriented from u to v, we write $u \to v$.

In the following proofs we will combine two flows into a new one as follows. Let (o_1, f_1) and (o_2, f_2) be k-flows on subgraphs G_1 and G_2 of a graph G, respectively. For $i \in \{1, 2\}$ we extend the flow (o_i, f_i) to flows on G, still called (o_i, f_i), by setting the value of f_i to be 0 and setting the orientation o_i arbitrarily for the edges not in G_i. For those edges e of G where $o_1(e) \neq o_2(e)$, we change both the orientation of $o_2(e)$ and the value $f_2(e)$ to $-f_2(e)$ thereby transforming (o_2, f_2) to a flow (o_1, f_2'). The *combination* of flows (o_1, f_1) and (o_2, f_2) is the flow $(o_1, f_1 + f_2')$ on G. Transforming this obtained flow to an all-positive flow, we get a flow (o, f) on G, which we call the *positive combination* of flows (o_1, f_1) and (o_2, f_2).

A *smooth orientation* of a set of edge-disjoint cycles is an orientation such that for every cycle in the set, one edge is incoming and one edge is outgoing at every vertex in the cycle.

Let G_1 and G_2 be two subgraphs of G and let (G_1, o_1) and (G_2, o_2) be two orientations. Let $Z \subseteq E(G_1) \cap E(G_2)$. We say that (G_1, o_1) and (G_2, o_2) are *consistent* on Z if o_1 and o_2 agree on all the edges from Z.

4.2 Theoretical results

Let (o, f) be an all-positive nowhere-zero k-flow on a cubic graph G. An edge e with $f(e) = 2$ is called a *strong* 2*-edge* if G has no 3-edge-cut containing the edge e such that the remaining edges of the cut have value 1 in f.

Lemma 4.5. *Let G be a 3-edge-connected graph and let (o, f) be an all-positive nowhere-zero k-flow on G for some integer k. Then all edges of G which receive value 1 and all strong 2-edges in (o, f) are deletable in o.*

Proof. Let e be an edge with $f(e) = 1$. Suppose that there exist two vertices of G, say u and v, such that there is no oriented uv-path in $(G - e, o)$. Let W be the set of vertices of G to which there exists an oriented path from u in $(G - e, o)$. Obviously, we have $u \in W$ and $v \notin W$. Let $W' = V(G) - W$. Let us look at the edge-cut S of $G - e$ between W and W'. All the edges in S must be oriented from W' to W, otherwise for some vertex from W' there would exist an oriented path from u to this vertex. Moreover, as G is 3-edge-connected, we have $|S| \geq 2$.

Now consider the edge-cut S^* between W and W' in G; either $S^* = S$ or $S^* = S \cup \{e\}$. Recall that (o, f) is an all-positive nowhere-zero flow. Since (o, f) is a flow, it holds that on any edge-cut the sum of the values on the edges oriented in one direction equals the sum of the values on the edges oriented in the other direction. Since all the edges of S are oriented in the same direction and have non-zero value, it cannot happen that $S^* = S$. So it must be the case that $S^* = S \cup \{e\}$ and all the edges of S are oriented in the same direction and e is oriented in the opposite orientation. But $f(e) = 1$ and since $|S| \geq 2$ and all the values of f on the edges of S are positive, this cannot happen either. Therefore we conclude that for any two vertices u and v there exists an oriented uv-path in $(G - e, o)$ and so e is deletable.

Assume now that e is a strong 2-edge and suppose that e is not deletable. We define the cuts S in $G - e$ and S^* in G similarly as above. Since f is all-positive and G is 3-edge-connected, S^* has to contain exactly three edges, e and two edges in S oriented oppositely from e and valuated 1. But this is impossible since e is a strong 2-edge. Therefore e is deletable. □

4.2.1 The Frank number of graphs with a nowhere-zero 4-flow

In this section, we prove Theorem 4.4. In the proof we utilize Lemma 4.5 and carefully apply the fact that every nowhere-zero 4-flow can be expressed as a combination of two 2-flows.

Proof of Theorem 4.4. Since G admits a nowhere-zero 4-flow, it also admits a nowhere-zero $(\mathbb{Z}_2 \times \mathbb{Z}_2)$-flow by a famous result of Tutte [132]. Let us denote by A the set of edges with value $(0,1)$, by B the set of edges with value $(1,0)$ and by C the set of edges with value $(1,1)$ in this flow. Note that by the nowhere-zero property there are no edges with value $(0,0)$.

Consider the subgraph G_1 of G induced by $A \cup C$. Since their edges all had a flow value with a 1 in the first coordinate and Kirchhoff's law holds around every vertex in G, G_1 is *Eulerian*, i.e. the degree of every vertex of G_1 is even. Therefore, G_1 consists of edge-disjoint cycles. Note that a vertex can belong to more than one cycle. Similarly, the subgraph G_2 induced by $B \cup C$ is Eulerian and so consists of edge-disjoint cycles.

Now fix a smooth orientation (G_1, o_1) of the cycles in G_1 and a smooth orientation (G_2, o_2) of the cycles in G_2. Set the value f_i to be i for the edges lying in G_i. Denote by (o, f) the positive combination of the flows (o_1, f_1) and (o_2, f_2). The value 1 in (o, f) is on all the edges of A and on those edges of C that have different orientation in (G_1, o_1) and (G_2, o_2).

Now we construct a complementary all-positive nowhere-zero 4-flow on G in a sense that this flow will have value 1 exactly on those edges where (o, f) had not. For $i \in \{1, 2\}$ we set $o'_i = o_i$ on G_i. We set $f'_1(e) = 2$ if $e \in G_1$ and $f'_2(e) = -1$ if $e \in G_2$. We create a flow (o', f') of G as the positive combination of the flows (o'_1, f'_1) and (o'_2, f'_2).

Summing up, we have constructed two all-positive nowhere-zero 4-flows on G, namely flows (o, f) and (o', f'). The edges of A are valuated 1 in (o, f). The edges of B are valuated 1 in (o', f'). The edges e of C where $o_1(e) \neq o_2(e)$ are valuated 1 in (o, f). The edges e of C where $o_1(e) = o_2(e)$ are valuated 1 in (o', f'). Therefore, each edge has value 1 either in (o, f) or in (o', f') and by Lemma 4.5 we have that $fn(G) = 2$.

It is known that a cubic graph is 3-edge-colourable if and only if it admits a nowhere-zero 4-flow. Therefore the second part of the theorem follows. □

Since every hamiltonian cubic graph is 3-edge-colourable, we have also shown the following conjecture by Barát and Blázsik [11].

Corollary 4.6. *If G is a 3-edge-connected cubic graph admitting a hamiltonian cycle, then $fn(G) = 2$.*

4.2.2 A general upper bound for the Frank number

In this section we prove Theorem 4.2 and thereby improve the previous general upper bound by Hörsch and Szigeti from 7 to 4. We note that in [10] Barát and Blázsik recently independently proved that $fn(G) \leq 5$. As a main tool we again use Lemma 4.5.

Proof of Theorem 4.2. By a result of Barát and Blázsik it is sufficient to prove the theorem for 3-edge-connected cubic graphs.

Let G be a 3-edge-connected cubic graph. By Seymour's 6-flow theorem [119], G has a nowhere-zero $(\mathbb{Z}_2 \times \mathbb{Z}_3)$-flow (o, f). The edges $e \in E(G)$ for which $f(e)$ is non-zero in the first coordinate induce a subgraph D. As every vertex in G can either have none or two of such edges, D is a set of vertex-disjoint cycles. The edges $e \in E(G)$ for which $f(e)$ is non-zero in the second coordinate induce a subgraph H'. As H' admits a nowhere-zero 3-flow there are no vertices of degree 1 in H', hence H' consists of a set of vertex-disjoint cycles and a subdivision of a cubic graph H. The graph H is bipartite since it is cubic and has a nowhere-zero 3-flow.

We will create four all-positive nowhere-zero k-flows on G using the subgraphs D and H' such that each edge is valuated 1 in at least one of the flows. Then Lemma 4.5 will imply the result.

Since H is cubic and bipartite, by Kőnig's line colouring theorem [90], it is 3-edge-colourable. Hence, it admits a proper edge-colouring with colours a, b, c. Fixing such a colouring φ, we find a (not-necessarily proper) edge-colouring φ' on H' as follows. If an edge e in H corresponds to a path P in H', we colour all the edges of P in H' by $\varphi(e)$. All the edges lying on cycles in H' will receive the colour a in φ'. Denote the set of edges with colour a, b, or c in φ' in H' by A, B, or C, respectively.

Now fix a smooth orientation (D, o_D) of the cycles in D and an orientation of H by directing all edges from one partite set to the other. We can find an orientation $(H', o_{H'})$ of H' as follows. Each oriented edge in H will correspond to an oriented path in H', oriented in the same direction. We take any smooth orientation of the cycles of H'. This fixes an orientation $(H', o_{H'})$ of H'.

We now partition the edges of G based on the orientations and colors of D and H'. Denote by D_0 the set of edges which lie only in D. Denote by A_0 the set of edges which lie only in A, by A_+ the set of edges e lying in A and D such that $o_{H'}$ and o_D have the same direction for e and by A_- the set of edges e in A and D such that $o_{H'}$ and o_D direct e oppositely. Similarly, we define B_0, B_+, B_- and C_0, C_+ and C_-. It is easy to check that every edge belongs to exactly one of D_0, A_0, A_+, A_-, B_0, B_+, B_-, C_0, C_+, and C_-.

Each of the four nowhere-zero flows (o_i, h_i) for $i \in \{1, 2, 3, 4\}$ will be the positive combination of a flow on D and a flow on H'. In each of the four cases cases we proceed as follows.

We define flows $(o_{i,1}, g_{i,1})$ on D for $i \in \{1, 2, 3, 4\}$. For edges $e \in E(D)$, let $o_{i,1}(e) = o_D(e)$ and $g_{i,1}(e) = g_{i,D}$ where $g_{i,D}$ is the value according to Table 4.1.

We define flows $(o_{i,2}, g_{i,2})$ on H' for $i \in \{1, 2, 3, 4\}$. For edges $e \in E(H')$, let $o_{i,2}(e) = o_{H'}(e)$ and let $g_{i,2}(e)$ equal $g_{i,A}$, $g_{i,B}$ or $g_{i,C}$ if e is in A, B or C, respectively, where $g_{i,A}$, $g_{i,B}$, and $g_{i,C}$ are three values that sum to 0, according to Table 4.1.

The flow (o_i, h_i) will be the positive combination of $(o_{i,1}, g_{i,1})$ and $(o_{i,2}, g_{i,2})$. For each $i \in \{1, 2, 3, 4\}$ the flow values are given in Table 4.1.

Since for each $i \in \{1, 2, 3, 4\}$, the sum of $g_{i,A}$, $g_{i,B}$ and $g_{i,C}$ is zero, it is easy to see that (o_i, h_i) is a k-flow. Moreover, as $g_{i,D}$, $g_{i,A}$, $g_{i,B}$ and $g_{i,C}$ are non-zero and $|g_{i,D}|$ differs from $|g_{i,A}|$, $|g_{i,B}|$ or $|g_{i,C}|$, we see that each (o_i, h_i) is nowhere-zero.

Finally, one can see that for every edge e, $h_i(e)$ is 1 for at least one i. The result follows. □

Flow	$g_{i,D}$	$g_{i,A}$	$g_{i,B}$	$g_{i,C}$	$h_i(e) = 1$
(o_1, h_1)	1	2	2	-4	$e \in D_0 \cup A_- \cup B_-$
(o_2, h_2)	3	1	1	-2	$e \in A_0 \cup B_0 \cup C_+$
(o_3, h_3)	2	3	-4	1	$e \in C_0 \cup A_- \cup C_-$
(o_4, h_4)	2	-3	-1	4	$e \in A_+ \cup B_+ \cup B_0$

Table 4.1: Four nowhere-zero k-flows defined by the procedure described in the proof of Theorem 4.2. Each (o_i, h_i) is the positive combination of a flow on D having value $g_{i,D}$ on each edge and a flow on H', whose edge set can be partitioned into sets A, B and C, having value $g_{i,A}$ on edges of A, $g_{i,B}$ on edges of B and $g_{i,C}$ on edges of C. These values are found on the i'th row in their respective column. The final column indicates which edges obtain value 1 in each of the positive combinations.

4.2.3 Reduction to cubic graphs

Barát and Blázsik showed in [11] that for any 3-edge-connected graph G, there exists a 3-edge-connected cubic graph H with $fn(H) \geq fn(G)$. Using the notion of *local cubic modification*, i.e. replace a vertex v of degree $d \geq 3$ with a cycles $v_1v_2 \dots v_d$ and replace each edge vx_i, where $x_1, \dots, x_d$ are the neighbours of v with an edge v_jx_i such that every v_i has degree 3. Starting with a graph G and applying this operation gives the local cubic modification G_v of G at v. Note that G_v is not unique and depends on the perfect matching chosen between $v_1, \dots, v_d$ and $x_1, \dots, x_d$.

We extend this result to cyclically 4-edge-connected 3-edge-connected graphs.

Lemma 4.7. *For any cyclically* 4*-edge-connected* 3*-edge-connected graph G and an arbitrary vertex $v \in V(G)$ of degree at least* 4*. There exists a local cubic modification G_v of G at v such that G_v is cyclically* 4*-edge-connected and* 3*-edge-connected.*

Proof. Fix a local cubic modification G_v of G. If G_v contains a bridge, then we have a bridge in G, so G_v is 2-edge-connected. Barát and Blázsik showed in [11, Lemma 4.1] that a 2-edge-cut must intersect $C_v = v_1 \dots, v_d$ twice and hence v is a cut vertex of G.

Now suppose that G_v contains a cyclic 3-edge-cut X. We show that in this case X must also intersect the cycle $C_v = v_1 \dots, v_d$ exactly twice. A cut must intersect C_v an even number of times, so suppose that X does not intersect C_v. If $G_v - X$ has cycles which are not C_v in each component, then X is also a cyclic 3-edge-cut of G, which is a contradiction. Hence, one of the components of $G_v - X$ can only have C_v as a cycle. This corresponds to an acyclic component of $G - X$ in which all but the vertices incident with X in G have degree at least 3, since G is 3-edge-connected. However, this can only happen if the acyclic component is the single vertex v, in contradiction with the fact that v has degree at least 4. Hence, a cyclic 3-edge-cut in G_v for any choice of perfect matching must intersect C_v twice. We conclude that if G_v contains a 2-edge cut or a cycle separating 3-edge cut X, the cut X is intersected by C_v in two edges.

We see that $G - v$, which is not necessarily connected, consists of 2-edge-connected components or single vertices which are connected to other components by bridges of $G - v$. We label all such components connected to at most one other via a bridge in $G - v$ by $K_1, \dots, K_k$. Note that by 3-edge-connectivity for any K_i, we have $|V(K_i) \cap N_G(v)| \geq 2$.

We now construct a perfect matching such that no two or three edges of G_v define a 2-edge-cut or a cyclic 3-edge-cut of G_v, respectively. By the previous

remark, we have $d \geq 2k$. For $i \in \{1, \ldots, k\}$, connect v_i with a vertex of K_i, for $i \in \{k+1, \ldots, 2k\}$, connect v_i with a vertex of K_{i-k}, connect the remaining vertices of C_v arbitrarily to the remaining neighbours of v in G. Let X be a 2-edge-cut or a cyclic 3-edge-cut of G_v. Then it must separate a vertex x in some K_{i_1} from a vertex y in some K_{i_2}, with $i_1 < i_2$. Clearly, x is connected to v_{i_1}, so one edge of X is $v_{i'_1}v_{i'_1+1}$ with $i_1 \leq i'_1 < i'_1 + 1 \leq i_2$. Otherwise, x and y are are in the same component of $G_v - X$. Similarly, the other edge of X must be $v_{i'_2}v_{i'_2+1}$ with $i_2 \leq i'_2 < i'_2 + 1 \leq k + i_1$. However, v_{k+i_1} is still connected to v_{k+i_2} on C_v, hence x and y are connected. □

Using Barát and Blázsik's Lemma 4.3 from [11], we obtain the following corollary.

Corollary 4.8. *Let G be a cyclically 4-edge-connected 3-edge-connected graph, then there exists a cyclically 4-edge-connected cubic graph H with $fn(H) \geq fn(G)$.*

4.2.4 Sufficient conditions for Frank number 2

The following lemmas and theorems give two sufficient conditions for a cyclically 4-edge-connected cubic graph to have Frank number 2. These will be used in the algorithm in Section 4.3. We note that for the conditions to hold the graphs need to have a 2-factor with exactly two odd cycles. For the graphs we consider in Section 4.3, the vast majority will have such a 2-factor. Even though the ideas we use can possibly be extended to 2-factors with a higher number of odd cycles, the proofs will be more involved and they will yield little speedup for our computations.

In the proof of the next lemma we will use the following proposition which can be found in [11] as Proposition 2.2.

Proposition 4.9. *Let (G, o) be a strong orientation of a graph G. Assume that an edge $e = uv$ is oriented from u to v in (G, o). The edge e is deletable in (G, o) if and only if there exists an oriented uv-path in $(G - e, o)$.*

Lemma 4.10. *Let (G, o) be a strong orientation of a cubic graph G. Let $e_1 = u_1v_1$ and $e_2 = u_2v_2$ be two non-adjacent edges in G such that (G, o) contains $u_1 \to v_1$ and $u_2 \to v_2$. Assume that both e_1 and e_2 are deletable in (G, o). Create a cubic graph G' from G by subdividing the edges e_1 and e_2 with vertices x_1 and x_2, respectively, and adding a new edge between x_1 and x_2. Let (G', o') be the orientation of G' containing $u_1 \to x_1$, $x_1 \to v_1$, $x_1 \to x_2$, $u_2 \to x_2$, $x_2 \to v_2$ and such that $o'(e) = o(e)$ for all the remaining edges of G'. Then*

$$D(G', o') \supseteq (D(G, o) - \{e_1, e_2\}) \cup \{x_1v_1, x_1x_2, u_2x_2\}.$$

Proof. First of all, define (G', o') as in the statement of this Lemma. Then it is a strong orientation. Indeed, since (G, o) is a strong orientation of G, any edge-cut in G contains edges in both directions in (G, o). Therefore, any edge-cut in G' contains edges in both directions in (G', o').

Now we are going to show that

$$D(G', o') \supseteq (D(G, o) - \{e_1, e_2\}) \cup \{x_1v_1, x_1x_2, u_2x_2\}.$$

Let $e = pr \in D(G, o) - \{e_1, e_2\}$ and let $p \to r$ be an oriented edge in (G, o). By Proposition 4.9 we have to show that there is an oriented pr-path in $(G' - e, o')$. If neither of p and r belongs to $\{x_1, x_2\}$, that is p and r belong to $V(G)$, then, since (G, o) is a strong orientation of G, there exists an oriented pr-path R in (G, o). Then R with possible subdivisions by x_1 and x_2 if $e_1 \in R$ or $e_2 \in R$ is the required pr-path in $(G' - e, o')$.

If $pr = x_1v_1$, then we find a v_2v_1-path R_1 in $(G - e_1, o)$, which exists since e_1 is deletable in o. The path $x_1x_2v_2R_1$ is an oriented x_1v_1-path in $(G' - x_1v_1, o')$.

If $pr = u_2x_2$, then we find a u_2u_1-path R_2 in $(G - e_2, o)$, which exists since e_2 is deletable in o. The path $R_2u_1x_1x_2$ is an oriented u_2x_2-path in $(G' - u_2x_2, o')$.

Finally, if $pr = x_1x_2$, we find a v_1u_2-path R_3 in (G, o). The path $x_1v_1R_3u_2x_2$ is an oriented x_1x_2-path in $(G' - x_1x_2, o')$. □

Theorem 4.11. *Let G be a cyclically 4-edge-connected cubic graph. Let C be a 2-factor of G with exactly two odd cycles, say N_1 and N_2 (and possibly some even cycles). Let $e = x_1x_2$ be an edge of G such that $x_1 \in V(N_1)$ and $x_2 \in V(N_2)$. Let $F = G - C$ and let M be a maximum matching in $C - \{x_1, x_2\}$. If there exists a smooth orientation of the cycles in $F - \{e\} \cup M$ and a smooth orientation of C such that each is consistent on the edges of N_i at distance 1 from x_i for both $i \in \{1, 2\}$, see Figure 4.1, then $fn(G) = 2$.*

Proof. For $i \in \{1, 2\}$ denote by u_i and v_i the vertices of N_i which are adjacent to x_i and let the rest of notation be as in the statement of the theorem. Consider the graph $G \sim e$, that is the graph created by deleting e and *smoothing* the two 2-valent vertices, i.e. removing them and adding an edge between their neighbours. First, we observe that $G \sim e$ is cyclically 3-edge-connected. This can be easily seen, because if $G \sim e$ had a cycle-separating k-edge-cut S for some $k < 3$, then $S \cup \{e\}$ would be a set of $k + 1$ edges separating two components containing cycles, so the cyclic connectivity of G would be at most $k + 1 < 4$, a contradiction.

Let $(F - \{e\} \cup M, o_2)$ be a smooth orientation of the cycles in $F - \{e\} \cup M$ consistent on the edges of N_i which are at distance 1 from x_i for both $i \in \{1, 2\}$.

Let (C, o_1) be such a smooth orientation of the cycles in C that the edges of $N_i \cap M$ incident with u_i and v_i on N_i are oriented equally in (C, o_1) and $(F - \{e\} \cup M, o_2)$. By our assumption these orientation exists. With slight abuse of notation, we will also consider C and $F - \{e\} \cup M$ to be subgraphs of $G \sim e$ and consider (C, o_1) and $(F - \{e\} \cup M, o_2)$ to be orientations of these subgraphs.

We define two nowhere-zero 4-flows (o', f') and (o'', f'') on $G \sim e$ such that $D(G \sim e, o') \cup D(G \sim e, o'') = E(G \sim e)$ and $\{u_1v_1, u_2v_2\} \subset D(G \sim e, o') \cap D(G \sim e, o'')$. This will by Lemma 4.10 imply that $fn(G) = 2$.

We define flows (o_1, f_1') on C and (o_2, f_2') on $F - \{e\} \cup M$. For edges $d \in E(C)$, let $f_1'(d) = 1$. For edges $d \in E(F - \{e\} \cup M)$, let $f_2'(d) = -2$. The flow (o', f') on $G \sim e$ will be the positive combination of (o_1, f_1') and (o_2, f_2').

Similarly, we define flows (o_1, f_1'') on C and (o_2, f_2'') on $F - \{e\} \cup M$. For edges $d \in E(C)$, let $f_1''(d) = 2$. For edges $d \in E(F - e \cup M)$, let $f_2''(d) = 1$. The flow (o'', f'') on $G \sim e$ will be the positive combination of (o_1, f_1'') and (o_2, f_2'').

It is easy to see that both (o', f') and (o'', f'') are nowhere-zero. We also see that the edges of $C - M$ are valuated 1 in (o', f'), the edges of $F - \{e\}$ are valuated 1 in (o'', f'') and the edges of M are valuated 1 in (o', f') if o_1 and o_2 agree and are valuated 1 in (o'', f'') if o_1 and o_2 disagree. Therefore, by Lemma 4.5, all edges of $G \sim e$ are deletable in one of these orientations and so $D(G \sim e, o') \cup D(G \sim e, o'') = E(G \sim e)$.

It remains to show that the edges u_1v_1 and u_2v_2 are deletable both in $(G \sim e, o')$ and in $(G \sim e, o'')$. By Lemma 4.5, they are deletable in $(G \sim e, o')$. Now we show that u_1v_1 and u_2v_2 are strong 2-edges in (o'', f''). Suppose that, for some $i \in \{1, 2\}$, the edge u_iv_i is not a strong 2-edge in (o'', f''), say u_1v_1 is not a strong 2-edge. Since we have already observed that $G \sim e$ is cyclically 3-edge-connected, this implies that u_1v_1 belongs to a 3-edge-cut, say R, and the other two edges of this 3-edge-cut have to be valuated 1. The cut R must be cycle-separating. Otherwise, it would separate either u_1 or v_1 from the rest of the graph, contradicting the fact that the edges of M incident with u_1 and v_1 are valuated 3 in (o'', f'').

We will use the symbols N_1 and N_2 to denote also the cycles in $G \sim e$ corresponding to N_1 and N_2. Since every edge-cut intersects a cycle an even number of times, R contains two edges from N_1 (one of them is u_1v_1, let the other be g) and an edge $f \in E(F) \setminus \{e\}$. Let (V_1, V_2) be the partition of $V(G)$ corresponding to the cut R. All the vertices of the cycle N_2 belong either to V_1 or to V_2. In the former case, the partition $(V_1 \cup \{x_1, x_2\}, V_2)$ and in the latter case, the partition $(V_1, V_2 \cup \{x_1, x_2\})$ form a partition corresponding to a 3-edge-cut in G. As the two edges of this cut belonging to N_1 are independent

the cut is a cycle-separating 3-edge-cut in G, which is a contradiction. □

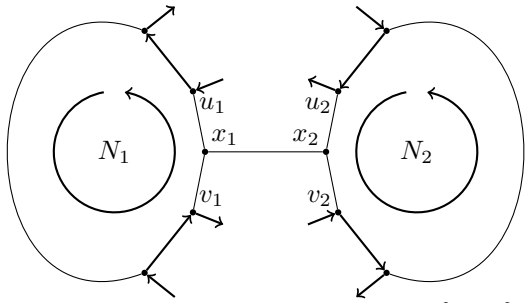


Figure 4.1: A smooth orientation of the cycles in $F - \{x_1x_2\} \cup M$ and of those in C such that each is consistent on the edges of N_i at distance 1 from x_i, for $i \in \{1, 2\}$.

Lemma 4.12. *Let (G, o) be a strong orientation of a cubic graph G. Let $e_1 = u_1v_1$, $e_2 = u_2v_2$, and $f = w_2w_1$ be pairwise independent edges in G such that (G, o) contains $u_1 \to v_1$, $u_2 \to v_2$, and $w_2 \to w_1$ and such that these edges are deletable in (G, o). Let a cubic graph G' be created from G by performing the following steps:*

- *subdivide the edges e_1 and e_2 with the vertices x_1 and x_2, respectively,*
- *subdivide the edge w_1w_2 with the vertices y_1 and y_2 (in this order), and*
- *add the edges x_1y_1 and x_2y_2.*

Let (G', o') be the orientation of G' containing $u_1 \to x_1$, $x_1 \to v_1$, $y_1 \to w_1$, $y_2 \to y_1$, $w_2 \to y_2$, $u_2 \to x_2$, $x_2 \to v_2$ and such that $o'(e) = o(e)$ for all the remaining edges of G' except for x_1y_1 and x_2y_2. Then

(a) if (G', o') contains $y_1 \to x_1$ and $x_2 \to y_2$, (G', o') is a strong orientation of G' and

$$D(G', o') \supseteq (D(G, o) - \{e_1, e_2, f\}) \cup \{u_1x_1, x_1y_1, y_1w_1, y_2w_2, x_2y_2, x_2v_2\}$$

(Figure 4.2(left)*);*

(b) if (G', o') contains $x_1 \to y_1$ and $y_2 \to x_2$, (G', o') is a strong orientation of G' and

$$D(G', o') \supseteq (D(G, o) - \{e_1, e_2, f\}) \cup \{x_1v_1, y_1y_2, u_2x_2\}$$

(Figure 4.2(right)*).*

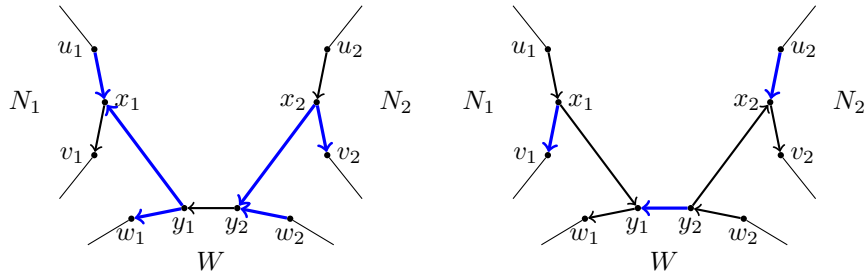


Figure 4.2: A part of G' and orientation (G', o') as defined in Lemma 4.12. The left-hand-side corresponds with the orientation of (a) and the right-hand-side corresponds with the orientation of (b). If the conditions of Lemma 4.12 are met the thick, blue edges will be deletable.

Proof. First of all, if (G', o') is defined either as in (a) or as in (b), then it is a strong orientation. Indeed, since (G, o) is a strong orientation of G, any edge-cut in G contains edges in both directions in (G, o). Therefore, any edge-cut in G' contains edges in both directions in (G', o').

To prove (a) we need to show that every edge from

$$(D(G, o) - \{e_1, e_2, f\}) \cup \{u_1x_1, x_1y_1, y_1w_1, y_2w_2, x_2y_2, x_2v_2\}$$

is deletable in (G', o'). To do so, it is enough to show that for any edge $e = pr$ in this set oriented as $p \to r$, there exists an oriented pr-path in $(G' - e, o')$. If $e \in D(G, o) - \{e_1, e_2, f\}$, then we can use the fact that e is a deletable edge in (G, o) and therefore there is an oriented pr-path in $(G-e, o)$. The corresponding pr-path in (G', o') possibly subdivided by some of the vertices x_1, x_2, y_1, y_2 is a required path.

If $pr = u_1x_1$, we take a u_1w_2-path R_1 in $G - e_1$ (it exists since e_1 is deletable in (G, o)). Possibly subdivide R_1 to R_1' in (G', o'). The path $R_1'w_2y_2y_1x_1$ is an oriented u_1x_1-path in $(G' - u_1x_1, o')$, so u_1x_1 is deletable in (G', o').

We proceed in this way. For every of the five remaining edges we find an oriented path R in (G, o) in which possibly one of the edges e_1, e_2, f is forbidden (such a path exists as each of these three edges is deletable in (G, o)), possibly subdivide R to R' in (G', o') and finally find the required oriented path T in $(G' - e, o')$. It is summarised in the following table.

e	R	T
u_1x_1	u_1w_2-path in $G - e_1$	$R'w_2y_2y_1x_1$
y_1x_1	w_1u_1-path in G	$y_1w_1R'u_1x_1$
y_1w_1	v_1w_1-path in $G - f$	$y_1x_1v_1R'$
w_2y_2	w_2u_2-path in $G - f$	$R'u_2x_2y_2$
x_2y_2	v_2w_2-path in G	$x_2v_2R'w_2y_2$
x_2v_2	w_1v_2-path in $G - e_2$	$x_2y_2y_1w_1R'$

To prove (b) we proceed analogously as in (a). There are only three edges we have to pay special attention to. Each of the edges corresponds to a line of the following table. This completes the proof.

e	R	T
x_1v_1	w_1v_1-path in $G - e_1$	$x_1y_1w_1R'$
y_2y_1	v_2u_1-path in $G - f$	$y_2x_2v_2R'u_1x_1y_1$
u_2x_2	u_2w_2-path in $G - e_2$	$R'w_2y_2x_2$

□

Theorem 4.13. *Let G be a cyclically 4-edge-connected cubic graph with a 2-factor C containing precisely two odd cycles N_1 and N_2 and at least one even cycle W. Let x_1y_1, y_1y_2 and y_2x_2 be edges of G such that $x_1 \in V(N_1)$, $x_2 \in V(N_2)$ and $y_1, y_2 \in V(W)$. For $i \in \{1, 2\}$ denote by u_i and v_i the vertices of N_i which are adjacent to x_i and by w_i the the vertex in $W - \{y_1, y_2\}$ which is adjacent to y_i in W. Let $F = G - C$ and let M be a maximum matching in $C - \{x_1, y_1, y_2, x_2\}$. If there exists a smooth orientation of the cycles in $F - \{x_1y_1, x_2y_2\} \cup M$ and a smooth orientation of C such that each is consistent on the edges of N_i at distance 1 from x_i for both $i \in \{1, 2\}$ and on the edges of W at distance 1 from y_i but not incident with y_{3-i}, for $i \in \{1, 2\}$, see Figure 4.3, and $G \sim x_1y_1 \sim x_2y_2$ is cyclically 3-edge-connected and has no cycle-separating 3-edge-cut $\{e_1, e_2, e_3\}$ with $e_1 \in \{u_1v_1, u_2v_2, w_1w_2\}$ and $e_2, e_3 \in E(F - \{x_1y_1, x_2y_2\} \cup M)$, then $fn(G) = 2$.*

Proof. Let $f_1 = x_1y_1$, $f_2 = x_2y_2$ and $G' := G \sim f_1 \sim f_2$. We define nowhere-zero 4-flows (o', f') and (o'', f'') on G' similarly as in the proof of Theorem 4.11. We can use the arguments from the latter theorem. Since we assume that none of the edges u_1v_1, u_2v_2, w_1w_2 are in a cycle-separating 3-edge-cut with two edges in $F - \{f_1, f_2\} \cup M$ they are strong 2-edges in (o'', f'') and the result follows from Lemma 4.5 and Lemma 4.12. □

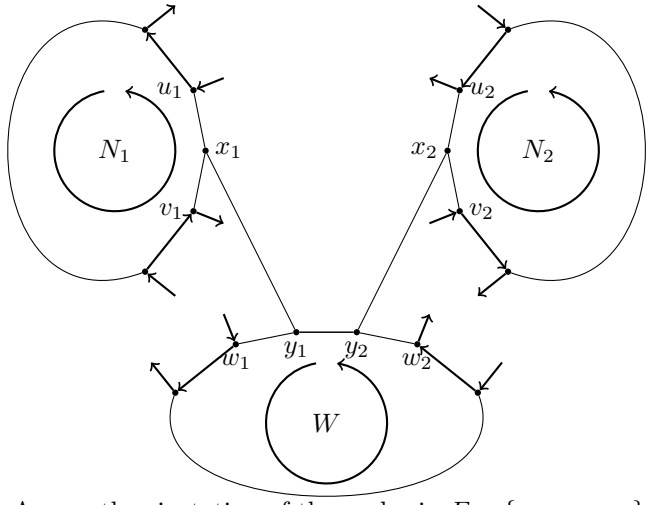


Figure 4.3: A smooth orientation of the cycles in $F - \{x_1y_1, y_2x_2\} \cup M$ and of the cycles in C such that each consistent on the edges of N_i at distance 1 from x_i, for $i \in \{1, 2\}$, and on the edges of W at distance 1 from y_i but not incident with y_{3-i}, for $i \in \{1, 2\}$.

4.3 Algorithm

We propose two algorithms for computationally verifying whether or not a given 3-edge-connected cubic graph has Frank number 2, i.e. a heuristic and an exact algorithm. Note that the Frank number for 3-edge-connected cubic graphs is always at least 2. Our algorithms are intended for graphs which are not 3-edge-colourable, since 3-edge-connected 3-edge-colourable graphs have Frank number 2 (cf. Theorem 4.4).

The first algorithm is a heuristic algorithm, which makes use of Theorem 4.11 and Theorem 4.13. Hence, it can only be used for cyclically 4-edge-connected cubic graphs. For every 2-factor in the input graph G, we verify if one of the configurations of these theorems is present. If that is the case, the graph has Frank number 2.

More specifically, we look at every 2-factor of G by generating every perfect matching and looking at its complement. We then count how many odd cycles there are in the 2-factor under investigation. If there are precisely two odd cycles, then we check for every edge connecting the two odd cycles whether or not the conditions of Theorem 4.11 hold. If they hold for one of these edges, we stop the algorithm and return that the graph has Frank number 2. If these conditions do not hold for any of these edges or if there are none, we check for

all triples of edges x_1y_1, y_1y_2, y_2x_2, where x_1 and x_2 lie on different odd cycles and y_1 and y_2 lie on the same even cycle of our 2-factor, whether the conditions of Theorem 4.13 hold. If they do, then G has Frank number 2 and we stop the algorithm. The pseudocode of this algorithm can be found in Algorithm 3.

Algorithm 3 heuristicForFrankNumber2(Graph G)

for each perfect matching F **do**
  Store odd cycles of $C := G - F$ in $\mathcal{O} = \{N_1, \ldots, N_k\}$
  if $|\mathcal{O}|$ is not 2 **then**
    Continue with the next perfect matching
  for all edges x_1x_2 with $x_1 \in V(N_1), x_2 \in V(N_2)$ **do**
    // Test if Theorem 4.11 can be applied
    Store a maximal matching of $C - \{x_1, x_2\}$ in M
    Denote the neighbours of x_1 and x_2 in C by u_1, v_1 and u_2, v_2, respectively
    Denote the set of edges of N_1 and N_2 at distance 1 from x_1 and x_2 by Z
    Create an empty partial orientation $(F - \{x_1, x_2\} \cup M, o)$
    for all $x \in \{u_1, v_1, u_2, v_2\}$ **do**
      if the cycle in $F - \{x_1, x_2\} \cup M$ containing x is not yet oriented **then**
        Orient the cycle in $F - \{x_1, x_2\} \cup M$ containing x such that it is smooth and maintaining consistency on Z with some orientation of C if possible.
    if $(F - \{x_1, x_2\} \cup M, o)$ can be extended to a smooth orientation consistent on Z with some smooth orientation of C **then**
      return True // Theorem 4.11 applies
  for all pairs of edges x_1y_1, x_2y_2 with $x_1 \in V(N_1), x_2 \in V(N_2)$ and y_1, y_2 adjacent and on the same even cycle W of C **do**
    // Test if Theorem 4.13 can be applied
    Store a maximal matching of $C - \{x_1, y_1, y_2, x_2\}$ in M
    Denote the neighbours of x_1 and x_2 in C by u_1, v_1 and u_2, v_2, respectively
    Denote the neighbour of y_1 in $C - y_2$ by w_1 and of y_2 in $C - y_1$ by w_2
    Denote the set of edges of N_i at distance 1 from x_i and of W at distance 1 from y_i but not incident with y_{3-i} by Z
    Create an empty partial orientation $(F - \{x_1, y_1, y_2, x_2\} \cup M, o)$
    for all $x \in \{u_1, v_1, u_2, v_2, w_1, w_2\}$ **do**
      if the cycle in $F - \{x_1, y_1, y_2, x_2\} \cup M$ with x is not oriented in (G, o) **then**
        Orient the cycle in $F - \{x_1, y_1, y_2, x_2\} \cup M$ containing x such that it is smooth and maintaining consistency on Z with some orientation of C
    if $(F - \{x_1, y_1, y_2, x_2\} \cup M, o)$ can be extended to a smooth orientation consistent on Z with some orientation of C **then**
      if $G \sim x_1y_1 \sim x_2y_2$ is not cyclically 3-edge-connected **then**
        Continue with for loop
      // Check cycle-separating edge-set condition
      for all pairs of edges e_1, e_2 in $F - \{x_1, y_1, y_2, x_2\} \cup M$ **do**
        for all $e \in \{u_1x_1, w_1y_1, u_2x_2\}$ **do**
          if $\{e, e_1, e_2\}$ cyclically separates $G - x_1y_1 - x_2y_2$ **then**
            return True // Theorem 4.13 applies
return False

Note that in practice, when checking the conditions of Theorem 4.11 and Theorem 4.13, we only consider one maximal matching as defined in the statements of the theorems. This has no effect on the correctness of the heuristic algorithm, but if we choose a matching for which the conditions of Theorem 4.11 and Theorem 4.13 do not hold, the heuristic will not be sufficient to decide whether or not the Frank number is 2. The second algorithm is then needed to decide this. As we will see in the results section, this approach is sufficient to generate all graphs in the relevant class up to all orders for which it is feasible to generate them.

The second algorithm is an exact algorithm for determining whether or not a 3-edge-connected cubic graph has Frank number 2. The pseudocode of this algorithm can be found in Algorithm 4. For a graph G, we start by considering each of its strong orientations (G, o) and try to find a complementary orientation (G, o') such that every edge is deletable in either (G, o) or (G, o'). First, we check if there is a vertex in G for which none of its adjacent edges are deletable in (G, o). If this is the case, there exists no complementary orientation as no orientation of a cubic graph can have three deletable edges incident to the same vertex. If (G, o) does not contain such a vertex, we look for a complementary orientation using some tricks to reduce the search space.

More precisely, we first we start with an empty *partial orientation*, i.e. a directed spanning subgraph of some orientation of G, and fix the orientation of some edge. Note that we do not need to consider the opposite orientation of this edge, since an orientation of a graph in which all arcs are reversed has the same set of deletable edges as the original orientation.

We then recursively orient edges of G that have not yet been oriented. After orienting an edge, the rules of Lemma 4.15 may enforce the orientation of edges which are not yet oriented. We orient them in this way before proceeding with the next edge. This heavily restricts the number edges which need to be added. As soon as a complementary orientation is found, we can stop the algorithm and return that the graph G has Frank number 2. If for all strong orientations of G no such complementary orientation is found, then the Frank number of G is higher than 2.

Since the heuristic algorithm is much faster than the exact algorithm, we will first apply the heuristic algorithm. After this we will apply the exact algorithm for those graphs for which the heuristic algorithm was unable to decide whether or not the Frank number is 2. In Section 4.3.1, we give more details on how many graphs pass this heuristic algorithm.

An implementation of these algorithms can be found on GitHub [43]. Our implementation uses bitvectors to store adjacency lists and lists of edges and

uses bitoperations to efficiently manipulate these lists.

Algorithm 4 frankNumberIs2(Graph G)

```
1: for all orientations (G, o) of G do
2:     if (G, o) is not strong then
3:         Continue with next orientation
4:     Store deletable edges of (G, o) in a set D
5:     for all v ∈ V(G) do
6:         if no edge incident to v is deletable then
7:             Continue with next orientation
8:     Create empty partial orientation (G, o') of G
9:     Choose an edge xy in G and fix orientation x → y in o'
10:    if not canAddArcsRecursively((G, o'), D, x → y) then // Algorithm 6
11:        Continue loop with next orientation
12:    if canCompleteOrientation((G, o'), D) then // Algorithm 5
13:        return True
14: return False
```

Algorithm 5 canCompleteOrientation(Partial Orientation (G, o'), Set D

```
1: if all edges are oriented in (G, o') then
2:     if D ∪ D(G, o') = E(G) then
3:         return True
4:     return False
5: // (G, o') still has unoriented edges
6: Store a copy of (G, o') in (G, o'')
7: Choose an edge uv unoriented in (G, o')
8: if canAddArcsRecursively((G, o'), D, u → v) then
9:     if canCompleteOrientation((G, o'), D) then
10:        return True
11: Reset o' using o''
12: if canAddArcsRecursively((G, o'), D, v → u) then
13:    if canCompleteOrientation((G, o'), D) then
14:        return True
15: return False
```

Theorem 4.14. *Let G be a cyclically 4-edge-connected cubic graph. If Algorithm 3 is applied to G and returns True, G has Frank number 2.*

Proof. Suppose the algorithm returns True for G. This happens in a specific iteration of the outer for-loop corresponding to a perfect matching F. The

complement of F is a 2-factor, say C, and since the algorithm returns True, C has precisely two odd cycles, say N_1 and N_2, and possibly some even cycles.

Suppose first that the algorithm returns True on Line 15. Then there is an edge x_1x_2 in G with $x_1 \in V(N_1)$ and $x_2 \in V(N_2)$, a maximal matching M of $C - \{x_1, x_2\}$ and orientations $(F - \{x_1x_2\} \cup M, o_1)$ and $(N_1 \cup N_2, o_2)$ which are consistent on the edges of N_i at distance 1 from x_i. Now by Theorem 4.11 it follows that G has Frank number 2.

Now suppose that the algorithm returns True on Line 33. Then there are edges x_1y_1, y_1y_2 and y_2x_2 such that $x_1 \in V(N_1)$, $x_2 \in V(N_2)$ and $y_1, y_2 \in V(W)$ where W is some even cycle in C. Since the algorithm returns True, there is a maximal matching M of $C - \{x_1, y_1, y_2, x_2\}$ and smooth orientations $(F - \{x_1x_2\} \cup M, o_1)$ and $(N_1 \cup N_2 \cup W, o_2)$ which are consistent on the edges of N_i at distance 1 from x_i and on the edges of W at distance 1 from y_i and are not incident with y_{3-i} for $i \in \{1, 2\}$.

Denote the neighbours of x_1 and x_2 in C by u_1, v_1 and u_2, v_2, respectively and denote the neighbour of y_1 in $C - y_2$ by w_1 and the neighbour of y_2 in $C - y_1$ by w_2. Since no triple e, e_1, e_2, where $e \in \{u_1x_1, w_1y_1, u_2x_2\}$, $e_1, e_2 \in E(F - \{x_1y_1, x_2y_2\} \cup M)$, is a cycle-separating edge-set of $G - \{x_1y_1, x_2y_2\}$, $G \sim x_1y_1 \sim x_2y_2$ has no cycle-separating edge-set $\{e, e_1, e_2\}$, where $e \in \{u_1v_1, u_2v_2, w_1w_2\}$ and $e_1, e_2 \in E(F - \{x_1y_1, x_2y_2\} \cup M)$. Now by Theorem 4.13 it follows that G has Frank number 2. □

We will use the following Lemma for the proof of the exact algorithm's correctness.

Lemma 4.15. *Let G be a cubic graph with $fn(G) = 2$ and let (G, o) and (G, o') be two orientations of G such that every edge $e \in E(G)$ is deletable in either (G, o) or (G, o'). Then the following hold for (G, o'):*

1. *every vertex has at least one incoming and one outgoing edge in (G, o'),*
2. *let $uv \notin D(G, o)$, then u has one incoming and one outgoing edge in $(G - uv, o')$,*
3. *let $uv, vw \notin D(G, o)$, then they are oriented either $u \to v$, $w \to v$ or $v \to u$, $v \to w$ in (G, o').*

Proof. We now prove each of the three properties:

1. Let u be a vertex such that all its incident edges are either outgoing or incoming in (G, o'). Clearly none of these edges can be deletable in (G, o'). Since there is some edge ux not in $D(G, o)$. We get a contradiction.

2. Let $uv \notin D(G, o)$ and let the remaining edges incident to u be either both outgoing or both incoming in (G, o'). Then uv is not deletable in (G, o') since all oriented paths to (respectively, from) u pass through uv.

3. Suppose without loss of generality that we have $u \to v$ and $v \to w$ in (G, o'). If the remaining edge incident to v is outgoing, then uv is not deletable in (G, o'). If the remaining edge is incoming, then vw is not deletable in (G, o'). □

Theorem 4.16. *Let G be a cubic graph. Algorithm 4 applied to G returns True if and only if G has Frank number 2.*

Proof. Suppose that frankNumberIs2(G) returns True. Then there exist two orientations (G, o) and (G, o') for which $D(G, o) \cup D(G, o') = E(G)$. Hence, $fn(G) = 2$.

Conversely, let $fn(G) = 2$. We will show that Algorithm 4 returns True. Let (G, o_1) and (G, o_2) be orientations of G such that every edge of G is deletable in either (G, o_1) or (G, o_2). Every iteration of the loop of Line 1, we consider an orientation of G. If the algorithm returns True before we consider (G, o_1) in this loop, the proof done. So without loss of generality, suppose we are in the iteration where (G, o_1) is the orientation under consideration in the loop of Line 1.

Without loss of generality assume that the orientation of xy we fix on Line 9 is in (G, o_2). (If not, reverse all edges of (G, o_2) to get an orientation with the same set of deletable edges.) Let (G, o') be a partial orientation of G and assume that all oriented edges correspond to (G, o_2). Let $u \to v$ be an arc in (G, o_2). If $u \to v$ is present in (G, o'), then canAddArcsRecursively(G, $D(G, o)$, o', $u \to v$) (Algorithm 6) returns True and no extra edges become oriented in (G, o'). If $u \to v$ is not present in (G, o'), it gets added on Line 8 of Algorithm 6, since the if-statement on Line 6 of Algorithm 6 will return True by Lemma 4.15. Note that this is the only place where an arc is added to (G, o') in Algorithm 6. Hence, if we only call Algorithm 6 on arcs present in (G, o_2), then all oriented edges e of (G, o') will always be oriented as $o_2(e)$. Now we will show that Algorithm 6 indeed only calls itself on arcs in (G, o_2).

Again, suppose $u \to v$ is an arc in (G, o_2), that it is not yet oriented in (G, o') and that every oriented edge e of (G, o') has orientation $o_2(e)$. The call canAddArcsRecursively(G, $D(G, o)$, o', $u \to v$) can only call itself on Line 9, i.e. in Algorithm 14. We show that in all cases after orienting uv as $u \to v$ in (G, o'), the call to Algorithm 6 only happens on arcs oriented as in (G, o_2).

Suppose u has two outgoing and no incoming arcs in (G, o'). Let ux be the final unoriented edge incident to u. Then (G, o_2) must have arc $x \to u$, otherwise it

has three outgoing arcs from the same vertex. Now suppose v has two incoming and no outgoing arcs in (G, o'). Let vx be the final unoriented edge incident to v. Then (G, o_2) must have arc $v \to x$, otherwise it has three incoming arcs to the same vertex.

Suppose uv is deletable in (G, o_1). Let ux also be deletable in (G, o_1). Denote the final edge incident to u by uy. Clearly, uy cannot be deletable in (G, o_1). Hence, it is deletable in (G, o_2). If (G, o_2) contains $u \to x$, then uy is not deletable in o_2. Hence, (G, o_2) contains $x \to u$. Let vx be a deletable edge of (G, o_1) and denote the final edge incident to v by vy. Since vy cannot be deletable in (G, o_1), (G, o_2) must contain arc $v \to x$. Suppose that the edges incident with u which are not uv are both not in $D(G, o_1)$. Then they must be oriented incoming to u in (G, o_2). Similarly, if the edges incident with v which are not uv are both not in $D(G, o_1)$, they must both be outgoing from v in (G, o_2).

Finally, suppose that uv is not a deletable edge in (G, o_1). Suppose that (G, o') still has one unoriented edge incident to u, say ux. If the other incident edges are one incoming and one outgoing from u, then (G, o_2) contains the arc $u \to x$. Otherwise, uv cannot be deletable in (G, o_2). Similarly, if (G, o') still has one unoriented edge incident to v, say vx and the remaining incident edges are one incoming and one outgoing, then the arc $x \to v$ must be present in (G, o_2). Otherwise, uv cannot be deletable in (G, o_2). If ux is not deletable in (G, o_1) $x \neq v$. Then (G, o_2) contains the arc $u \to x$. Otherwise, not both of uv and ux can be deletable in (G, o_2). Similarly, if vy is not deletable in (G, o_1) and $y \neq u$, then (G, o_2) must contain the arc $y \to v$. Otherwise, not both of uv and vy can be deletable in (G, o_2).

This shows that all calls of canAddArcsRecursively(G, $D(G, o)$, o', $u \to v$) to itself, where all oriented edges of (G, o') and $u \to v$ are oriented as in (G, o_2), have as the fourth parameter an arc oriented as in (G, o_2). Since, in Algorithm 5, we keep orienting edges until (G, o') is completely oriented and try to orient edge uv as $v \to u$ if (G, o') cannot be completed with $u \to v$, it follows by induction that unless Algorithm 4 returns True in some other case, it will return True when $(G, o') = (G, o_2)$. □

4.3.1 Results

Since by Theorem 4.4 all 3-edge-connected 3-edge-colourable (cubic) graphs have Frank number 2, in this section we will focus on *non*-3-edge-colourable cubic graphs, i.e. *snarks*.

Algorithm 6 canAddArcsRecursively(Partial Orientation (G, o'), Set D, Arc $u \to v$)

```
// Check if u → v can be added and recursively orient edges for which the orientation is enforced by the rules of Lemma 4.15
if u → v is present in (G, o') then
    return True
if v → u is present in (G, o') then
    return False
if adding u → v violates rules of Lemma 4.15 then // Algorithm 13 in Appendix C.1
    return False
Add u → v to (G, o')
if the orientation of edges enforced by Lemma 4.15 yields a contradiction then // Algorithm 14 in Appendix C.1
    return False
return True
```

In [18] Brinkmann et al. determined all cyclically 4-edge-connected snarks up to order 34 and those of girth at least 5 up to order 36. This was later extended with all cyclically 4-edge-connected snarks on 36 vertices as well [46]. These lists of snarks can be obtained from the House of Graphs [30] at: `https://houseofgraphs.org/meta-directory/snarks`. Using our implementation of Algorithms 3 and 4, we tested for all cyclically 4-edge-connected snarks up to 36 vertices if they have Frank number 2 or not. This led to the following result.

Proposition 4.17. *The Petersen graph is the only cyclically* 4*-edge-connected snark up to order* 36 *which has Frank number not equal to* 2*.*

This was done by first running our heuristic Algorithm 3 on these graphs. It turns out that there are few snarks in which neither the configuration of Theorem 4.11 nor the configuration of Theorem 4.13 are present. For example: for more than 99.97% of the cyclically 4-edge-connected snarks of order 36, Algorithm 3 is sufficient to determine that their Frank number is 2 (see Table C.1 in Appendix C.2 for more details). Thus we only had to run our exact Algorithm 4 (which is significantly slower than the heuristic) on the graphs for which our heuristic algorithm failed. In total about 214 CPU days of computation time was required to prove Proposition 4.17 using Algorithm 3 and 4 (see Table C.2 in Appendix C.2 for more details).

In [82] Jaeger defines a snark G to be a *strong snark* if for every edge $e \in E(G)$, $G \sim e$, i.e. the unique cubic graph such that $G - e$ is a subdivision of $G \sim e$, is not 3-edge-colourable. Hence, a strong snark containing a 2-factor which has

precisely two odd cycles, has no edge e connecting those two odd cycles, i.e. the configuration of Theorem 4.11 cannot be present. Therefore, they might be good candidates for having Frank number greater than 2.

In [18] it was determined that there are 7 strong snarks on 34 vertices having girth at least 5, 25 strong snarks on 36 vertices having girth at least 5 and no strong snarks of girth at least 5 of smaller order. By Proposition 4.17, their Frank number is 2. In [17] it was determined that there are at least 298 strong snarks on 38 vertices having girth at least 5 and the authors of [17] speculate that this is the complete set. We found the following.

Observation 4.18. *The* 298 *strong snarks of order* 38 *determined in [17] have Frank number* 2*.*

These snarks can be obtained from the House of Graphs [30] by searching for the keywords “strong snark".

The configurations of Theorem 4.11 and Theorem 4.13 also cannot occur in snarks of *oddness* 4, i.e. the smallest number of odd cycles in a 2-factor of the graph is 4. Hence, these may also seem to be good candidates for having Frank number greater than 2. In [46, 47] it was determined that the smallest snarks of girth at least 5 with oddness 4 and cyclic edge-connectivity 4 have order 44 and that there are precisely 31 such graphs of this order. We tested each of these and found the following.

Observation 4.19. *Let G be a snark of girth at least* 5*, oddness* 4*, cyclic edge-connectivity* 4 *and order* 44*. Then $fn(G) = 2$.*

These snarks of oddness 4 can be obtained from the House of Graphs [30] at `https://houseofgraphs.org/meta-directory/snarks`.

4.3.2 Correctness Testing

The correctness of our algorithm was shown in Theorem 4.14 and Theorem 4.16. We also performed several tests to verify that our implementations are correct.

Hörsch and Szigeti proved in [79] that the Petersen graph has Frank number 3. In [11] Barát and Blázsik showed that both Blanuša snarks and every flower snark has Frank number 2. We verified that for the Petersen graph both Algorithm 3 and Algorithm 4 give a negative result, confirming that its Frank number is larger than 2. For the Blanuša snarks and the flower snarks up to 40 vertices Algorithm 4 always shows the graph has Frank number 2 and the heuristic Algorithm 3 is able to show this for a subset of these graphs.

We also ran our implementation of Algorithm 4 on the cyclically 4-edge-connected snarks up to 30 vertices without running Algorithm 3 first. Results were in complete agreement with our earlier computation, i.e. Algorithm 4 independently confirmed that each of the snarks which have Frank number 2 according to Algorithm 3 indeed have Frank number 2. For the cyclically 4-edge-connected snarks on 30 vertices Algorithm 4 took approximately 46 hours. Our heuristic Algorithm 3 found the same results for all but 307 graphs in approximately 42 seconds.

During the computation of Algorithm 3 on the strong snarks, we verified it never returned True for the configuration of Theorem 4.11 and that it returned False for all snarks of oddness 4 mentioned earlier.

We also implemented a method for finding the actual orientations after Algorithm 3 detects one of the configurations. We checked for all cyclically 4-edge-connected snarks up to 32 vertices in which one of these configurations is found, whether the deletable edges for these two orientations form the whole edge set. This was always the case.

Another test was performed using a brute force algorithm which generates all strongly connected orientations of the graph and checks for every pair of these orientations whether the union of the deletable edges of this pair of orientations is the set of all edges. We were able to do this for all cyclically-4-edge-connected snarks up to 26 vertices and obtained the same results as with our other method. Note that this method is a lot slower than Algorithm 3 and 4. For order 26 this took approximately 152 hours, while using Algorithm 3 and 4 this took approximately 1 second.

Our implementation of Algorithm 3 and Algorithm 4 is open source and can be found on GitHub [43] where it can be inspected and used by others.

Acknowledgements

We would like to thank János Barát for useful discussions. The research of Edita Máčajová was partially supported by the grant No. APVV-19-0308 of the Slovak Research and Development Agency and the grant VEGA 1/0743/21. The research of Jan Goedgebeur and Jarne renders was supported by Internal Funds of KU Leuven. Several of the computations for this work were carried out using the supercomputer infrastructure provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government.

Chapter 5

Generation of Cycle Permutation Graphs and Permutation Snarks

This chapter is a minimally edited version of the following submitted work[1].

J. Goedgebeur, J. Renders and S. Van Overberghe. Generation of cycle permutation graphs and permutation snarks. *arXiv preprint*, 2025. arXiv: 2411.12606

My contributions to this paper, in accordance with the contributor role taxonomy (CRediT)[2], consist of: Conceptualisation, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft and Writing – Review and Editing.

5.1 Introduction

A *cycle permutation graph* is a cubic graph containing a 2-factor consisting of two chordless cycles. One can easily see that both of these cycles must have length $n/2$, where n is the order of the graph.

[1] An extended abstract of this work also appeared in the proceedings of the 50th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2025) [64].

[2] See https://credit.niso.org/.

Cycle permutation graphs were first introduced by Chartrand and Harary [25] in 1967. They characterised the conditions under which a cycle permutation graph is planar.

In 1972, Klee [89] also studied cycle permutation graphs, where he referred to them as "generalised prisms". He was interested in the hamiltonicity of cycle permutation graphs and asked for which orders there exist non-hamiltonian cycle permutation graphs. Using a computer search Klee determined that up to order 16, the Petersen graph is the only non-hamiltonian cycle permutation graph and he partially solved the question by proving that for every order $n \equiv 2 \bmod 4$ with $n \geq 18$ there exist non-hamiltonian cycle permutation graphs. In this manuscript, we finish this classification by completely characterising for which orders non-hamiltonian cycle permutation graphs exist.

Cycle permutation graphs were also studied in the 80's by Shawe-Taylor and Pisanski, who asked questions about the girth of such graphs [109, 110]. They construct cycle permutation graphs with an arbitrarily large girth.

We are also interested in cycle permutation graphs which are *snarks*, i.e. non-3-edge colourable. In this case we call them *permutation snarks*. We note that in some definitions, snarks are required to be *cyclically* 4*-edge-connected* and/or of *girth* at least 5, however, this is always the case for cycle permutation graphs of order $n > 6$. A graph is *cyclically* k*-edge-connected* if the removal of fewer than k edges cannot separate the graph into two components each containing a cycle and its *girth* is the length of a shortest cycle. Note that a cycle permutation graph with a cycle of length 4 is always hamiltonian and hamiltonian graphs cannot be snarks.

Snarks are particularly interesting since for a lot of open conjectures it can be shown that if the conjecture is false, the smallest possible counterexamples are snarks. This has amongst others been proven for the Cycle Double Cover conjecture [121, 125] and Tuttes 5-flow conjecture [132]. A better understanding of the class of snarks can therefore lead to a better understanding of many long-standing open conjectures. Permutation snarks in particular are of interest as their additional structure makes them natural candidates for investigation and this structure in turn allows us to study this class computationally up to significantly higher orders.

It is well known that the Petersen graph is a permutation snark. In 1997, Zhang conjectured that it is the only permutation snark which is cyclically 5-edge-connected [148]. This conjecture was refuted by Brinkmann, Goedgebeur, Hägglund and Markström [18] in 2013, when they developed a new generation algorithm for cubic graphs and snarks and generated all snarks up to 36 vertices and determined that there are 12 cyclically 5-edge-connected permutation snarks

on 34 vertices. They obtained counts for permutation snarks by using a filter approach, which given an input graph (given by the generation algorithm), determines whether or not it is a permutation snark. However, this method is not very efficient as in practice very few cubic graphs are cycle permutation graphs and even fewer are permutation snarks. For example on order 32 only 0.24% of the cubic graphs of girth at least 4 (the minimum girth of a cycle permutation graph of order > 6) are cycle permutation graphs. We improve the result by Brinkmann et al. by determining all permutation snarks up to order 46, yielding many more counterexamples to Zhang's conjecture.

In 2019, Máčajová and Škoviera [96] gave three methods for constructing permutation snarks and used them to provide permutation snarks of cyclic connectivity 4 and 5 for sufficiently large order $n \equiv 2 \pmod 8$. They mention that no permutation snark of order $n \equiv 6 \pmod 8$ is known (an open problem posed by Brinkmann et al. [18]) and prove that such a smallest permutation snark – if it exists – must by cyclically 5-edge-connected. Note that cycle permutation graphs of order $n \equiv 0 \pmod 4$ are trivially 3-edge-colourable since the two cycles in the 2-factor have even length.

We have developed two specialised algorithms for the exhaustive generation of all pairwise non-isomorphic cycle permutation graphs of a given order. The first algorithm is based on orderly generation [37, 112]. When interested in non-hamiltonian cycle permutation graphs or permutation snarks, we slightly adapt the algorithm in order to only take care of the most common isomorphisms. This speeds up the algorithm significantly at the cost of outputting some isomorphic copies of the same graph. We also describe a second algorithm using the canonical construction path method [99]. Its implementation is slower than the approach using orderly generation, however, we use it as a verification for the correctness of our implementation of the first algorithm. The orderly generation approach is described in Section 5.2, while the canonical construction path approach is described in Appendix D.1. Their implementations can be found on GitHub [52]. Variations of these algorithms allow to restrict the search to non-hamiltonian cycle permutation graphs, permutation snarks or graphs with a given lower bound on the girth in an efficient way. This allowed us to exhaustively generate all cycle permutation graphs and permutation snarks up to much higher orders.

The rest of this paper is organised as follows. In Section 5.2, we explain how the algorithms work and prove their correctness.

In Section 5.3, we use the algorithms to obtain counts for cycle permutation graphs as well as non-hamiltonian cycle permutation graphs and permutation snarks. For the latter, our algorithm allows us to generate all permutation snarks up to order 46. We also increase lower bounds for interesting permutation

snarks such as the minimum order for a permutation snark of order $n \equiv 6 \bmod 8$, which was also posed as an open problem in [18].

In Section 5.4, we finish the characterisation of the orders for which non-hamiltonian cycle permutation graphs exist, solving one of the questions by Klee [89].

5.2 Generation algorithms

In this section, we will present our algorithm for exhaustively generating all cycle permutation graphs of a given order n. The basic idea is a backtracking approach, where we start from two cycles of length $n/2$ connected by one edge and add edges between the cycles in all possible ways. Clearly, this generates all cycle permutation graphs. However, such an approach will give a lot of isomorphic copies. One way solve this is to keep track of all intermediate graphs and to stop adding edges whenever a graph isomorphic to a previously generated graph is found. However, this approach is very memory intensive. For example if one generates all cycle permutation graphs of order 28 using this approach, approximately 160 gigabytes of memory is needed. Therefore, we will use a different approach to avoid the generation of isomorphic copies.

In our algorithm (see Section 5.2.1), we reject isomorphisms using a method based on orderly generation [37, 112]. One labelling in each isomorphism class is determined to be (strongly) canonical and only these labellings will be accepted by the algorithm. Then every graph is generated precisely once without having to store any graphs in memory. For efficiency's sake, in the case we want to generate non-hamiltonian cycle permutation graphs or permutation snarks, we tolerate that some isomorphic graphs have multiple "canonical" labellings, which might lead to isomorphic graphs being output. (See Sections 5.2.2 and 5.2.3.) We call these labellings *weakly canonical*. While this method can output isomorphic copies of the same graph, this approach is a lot faster as we can avoid performing certain expensive computations while still filtering the most common isomorphs. Moreover, since the number of graphs output in this way remains relatively small, it is feasible to obtain the exact number of graphs by filtering the isomorphic copies in a post-processing step.

We now describe these algorithms in more detail and prove their correctness. A description of the algorithm using the canonical construction path method can be found in Appendix D.1. Our implementation of the algorithms can be found on GitHub [52] and in Appendix D.2 we describe how we extensively tested the correctness of the implementations in various ways.

5.2.1 Orderly generation method

The idea of this method is based on orderly generation, which was independently introduced by Faradžev [37] and Read [112] in 1978, but had already been used by Rozenfel'd [115] in 1973 for the generation of strongly regular graphs. Orderly generation defines a canonically labelled object for each isomorphism class and outputs only that one for each isomorphism class. As we will see, the problem of generating cycle permutation graphs lends itself well to orderly generation and is much faster than, for example, the canonical construction path method, which we describe in Appendix D.1.

We first define some terminology. Let G be a *subcubic* graph of even order n, i.e. a graph in which all vertices have degree at most 3. If G contains two vertex-disjoint chordless cycles C_1 and C_2 of length $k := n/2$, then $F := \{C_1, C_2\}$ is called a *permutation* 2*-factor* of G, i.e. a 2-factor consisting of chordless cycles C_1 and C_2. We call any edges of G that have one endpoint in C_1 and one endpoint in C_2 *spokes* of F. If the vertices of degree 2 in G on one of the cycles induce a path, we call this cycle *consecutive* and we call F a *consecutive permutation* 2*-factor* of G.

We will represent subgraphs of cycle permutation graphs in the following way. Let G be a subcubic graph containing a permutation 2-factor F with cycles C_1 and C_2. Let $C_1 = u_0u_1 \dots u_{k-1}$ and $C_2 = v_0v_1 \dots v_{k-1}$. Let $[x] := \{0, 1, \dots, x-1\}$ for the set of the first x integers[3]. Then G can be represented by the sequence $p = p_0, \dots, p_{k-1}$ (also known as a partial permutation), where p_i for $i \in [k]$ is given by

$$p : [k] \to [k] \cup \{?\} : i \mapsto \begin{cases} j, & \text{if } u_i \text{ is adjacent to } v_j, \\ ?, & \text{otherwise.} \end{cases}$$

We say p *represents* G via F. Note that G and F are not sufficient to determine p, we also need an assignment of the vertices of F to the variables u_i, v_i, with $i \in [k]$. Therefore, G can have many such sequences representing it even via the same permutation 2-factor F.

On the other hand, a sequence p gives rise to a labelling of G. Label the vertices of G from 0 to $n-1$ such that if $i \in [k]$, then it has neighbours, $i-1$, $i+1$, taking indices modulo k, and $k + p(i)$, if $p(i) \neq ?$. If $i \in [n] \setminus [k]$, then it has neighbours $i-1$, $i+1$, taking indices modulo k and then adding k, and j if there exists a $j \in [k]$ such that $p(j) = (i-k)$. In other words, if p represents G via F with cycles $u_0 \dots u_{k-1}$ and $v_0 \dots v_{k-1}$, this labelling is obtained from the isomorphism mapping u_i to i and v_i to $k+i$ for $i \in [k]$. We say this labelling, denoted by $G(p)$, is *given by* p.

[3]Note that this might deviate slightly from how this notation is typically used.

We call a sequence p *(strongly) canonical* if it is lexicographically smallest among all sequences representing G, where $i < ?$ for all $i \in [k]$. We call it *weakly canonical* if it is lexicographically smallest among all such sequences representing G via F (but as we will see later, being weakly canonical is independent of F). We will use weak canonicity in Section 5.2.2.

Algorithm 7 Expand(p, l)

```
if l = k then
    Output cubic graph represented by p.
for all x ∈ [k] \ im(p) do
    Define p' by p'(i) = p(i) for i ∈ [k] \ {l} and p'(l) = x.
    if p' is canonical then
        Expand(p', l + 1)
```

We now explain the algorithm in more detail. A rough outline of the recursive method without any optimisations or technical details can be found in Algorithm 7 by starting with $p = 0, ?, ?, \ldots, ?$ and $l = 1$. We denote the image of p by $\mathrm{im}(p)$.

Suppose we want to generate the cycle permutation graphs of order n. Let $k := n/2$. We start with G_0 consisting of two disjoint cycles $C_1 = u_0u_1 \ldots u_{k-1}$ and $C_2 = v_0v_1 \ldots v_{k-1}$ and the edge u_0v_0. Note that this is a subgraph of any cycle permutation graph. G_0 is then represented by $p^1 = 0, ?, ?, \ldots, ?$. Now let G be a (sub)cubic supergraph of G_0 of order n whose labelling is represented by p^l with $|\mathrm{im}(p^l) \setminus \{?\}| = l$. If G is cubic, then $k = l$ and we have a cycle permutation graph. It should be output and the recursion backtracks. If not, we recursively add edges from u_l to v_i with $i \in [k] \setminus \mathrm{im}(p)$. Adding such an edge u_lv_i gives us a new graph whose labelling is represented by a sequence $p' = p_0, \ldots, p_{l-1}, i, ?, \ldots, ?$. We must now decide to continue with p' if it is canonical, i.e. lexicographically minimal among all sequences representing G, or to discard it.

We now describe how to determine canonicity of a given sequence p representing a graph G. We can generate every sequence representing G using three operations on these sequences as long as we have such a sequence representing G via every permutation 2-factor F of G. We use the notation $\bmod\ k$ as a mapping sending an integer i_1 to an integer $i_2 \in \{0, \ldots, k-1\}$ such that $i_2 \equiv i_1 \bmod k$.

The first operation is a rotation $T : p \mapsto q$ such that $q(i) = p((i+1) \bmod k)$. It corresponds to an isomorphism ϕ_T from $G(p)$ to $G(q)$ sending vertices $i \in [k]$ to $(i+1) \bmod k$ and the remaining vertices to themselves. This can be thought of as rotating the labels of the first cycle. The second operation is a reflection $R : p \mapsto q$ such that $q(i) = p(k-1-i)$ for $i \in [k]$. It corresponds to an

isomorphism ϕ_R from $G(p)$ to $G(q)$ sending vertices $i \in [k]$ to $k-1-i$ and the remaining vertices to themselves. This can be seen as reflecting the first cycle. The third operation is an inversion, $I : p \mapsto q$ such that $q(i) = p^{-1}(i)$ if $i \in \mathrm{im}(p)$ and ? otherwise. It corresponds to an isomorphism ϕ_I from $G(p)$ to $G(q)$ sending the vertices $i \in [k]$ to $k+i$ and vice versa. This can be seen as swapping the cycles.

Lemma 5.1. *Let G be a (sub)cubic graph containing a permutation 2-factor. Then, any sequence representing G via a permutation 2-factor F can be obtained from any other sequence representing G via F by applying some composition of the above three operations T, R and I.*

Proof. Let p be a sequence representing G via F and let q be another such sequence. There exists an isomorphism ϕ between $G(p)$ and $G(q)$. We note that ϕ maps the cycles given by $0, \ldots, k-1$ and $k, \ldots, n-1$ to themselves or to each other. There are $2k$ ways a cycle can be mapped to another cycle, namely k choices to map the first element and a choice of direction. Since we are mapping two cycles and might swap these, this yields $8k^2$ options. It follows that ϕ is some composition of ϕ_T, ϕ_R and ϕ_I. We note that a rotation or reflection of the second cycle can be obtained by swapping them first. Since these isomorphisms correspond to T, R and I, the result follows. □

Therefore, in order to determine weak canonicity of a sequence p, it suffices to compare it to any other of the at most $8k^2$ sequences obtained by applying any composition of T, R and I to it. Hence, this does not depend on the given 2-factor via which it represents G.

We can now check (strong) canonicity of p as follows. For any permutation 2-factor F and a sequence p^F representing G via F, check if p is lexicographically smaller than all rotations of C_1 and or C_2 of F. Then check all rotations when the first cycle of F is reflected ($R(p^F)$), the second is reflected ($I \circ R \circ I(p^F)$), the cycles are swapped ($I(p^F)$) or any combination of these. Once a sequence smaller then p is found we can prune the search. We note that in practice, we use a slightly different notion of canonicity based on the difference lists of the sequences representing G and we use many optimisations such as not checking sequences for which it can immediately be determined that they are larger. We also use optimisations that allow us to check many of these sequences in constant time. See Appendix D.3 for some details on these optimisations.

For what follows and the proof of the algorithm's correctness we use the following two results.

Proposition 5.2. *Let G be a subcubic graph containing a permutation 2-factor. Every permutation 2-factor with cycles C_1 and C_2 will have the same number of degree 2 vertices in C_1 and C_2 (namely half of all degree 2 vertices in G).*

Proposition 5.3. *Let G be a subcubic graph containing a consecutive permutation 2-factor with cycles C_1 and C_2. Then every permutation 2-factor of G will necessarily be consecutive and partition the degree 2 vertices in the same way.*

Proof. Assume without loss of generality that C_1 is the cycle with consecutive degree 2 vertices (call this path P). Then any cycle C of a permutation 2-factor containing a vertex of P must contain all of P. But by Proposition 5.2 it cannot contain any further degree 2 vertices. Therefore C must also be consecutive. □

Note that the idea of orderly generation works in this case due to Proposition 5.3. If we reject a sequence p representing G via F because it is not canonical and the canonical sequence p' represents the graph via a different consecutive permutation 2-factor F', then F' will remain a 2-factor when extending p' due to this proposition.

In order to know if a given sequence representing G is canonical, we need to know all of G's permutation 2-factors. We keep track of this dynamically and since we checked this in the previous iteration of our algorithm, we only need to check for 2-factors containing the newly added edge. However, before starting the search for these 2-factors, we can first check the following condition.

Proposition 5.4. *Let G be a subcubic graph containing a consecutive permutation 2-factor with cycles C_1 and C_2 such that C_1 is consecutive. Let $e = uv$ be a non-edge of G with $u \in V(C_1)$ and $v \in V(C_2)$, such that C_1 is still consecutive in $G + e$. If v has a neighbour of degree 2, any consecutive permutation 2-factor of $G + e$ is also a consecutive permutation 2-factor of G.*

Proof. Let F be a consecutive permutation 2-factor of G with cycles C_1 and C_2. By construction, e is not a spoke with respect to F, so F is also a consecutive permutation 2-factor of $G + e$. By Proposition 5.3, a new permutation 2-factor F', if it exists, will have two cycles C_1' and C_2' such that, without loss of generality, all degree 2 vertices of C_1 belong to C_1' and all degree 2 vertices of C_2 belong to C_2'. Note that if a degree 2 vertex lies on a cycle, both of its neighbours must lie on the same cycle. If v has a neighbour of degree 2, then so must u (by Proposition 5.2). But one of those neighbours must be in C_1' and the other in C_2', which means that e cannot be part of a permutation 2-factor. □

If this condition holds, no new consecutive permutation 2-factors will be introduced by adding e to the graph. If the condition does not hold, we need to perform an exhaustive search for permutation 2-factors containing the new edge. For this, we use a backtracking algorithm for finding an induced cycle of length k. If G is not cubic, we initialise our cycle with the endpoints of the new edge and all vertices of the path, in which all interior vertices have degree 2, between one of these endpoints and u_0. If G is cubic, we can only initialise the cycle by the endpoints of the new edge. We then recursively extend this current path, by adding a neighbour of its end not introducing a chord. If this neighbour has another neighbour of degree 2, then we need not add it as the current cycle we are constructing is already saturated with degree 2 vertices, by Proposition 5.2. If the length of the path is less than k, but the ends do not have any good neighbours to add, we prune. If the length of the path is k and we have a cycle, then we check whether its complement is also a cycle. If this holds, then both cycles are induced and we have found a new permutation 2-factor.

Before proving the correctness of this algorithm, we note that a small variation allows for the generation of cycle permutation graphs with a given lower bound on the girth, namely by, before expanding the graph with a new edge, checking whether this new edge does not introduce a small cycle with length less than the given lower bound on the girth. We will use this variation to generate cycle permutation graphs with girth at least 5, 6,..., 9.

Let the restriction of p to its first l elements, denoted by $p|_l$, be defined by $p|_l(i) = p(i)$ if $i < l$ and $p|_l(i) = ?$ otherwise.

Theorem 5.5. *Algorithm 7 outputs every cycle permutation graph of a given order n exactly once.*

Proof. Suppose that the algorithm outputs at least two distinct sequences p and p' representing a cycle permutation graph G. Then one is lexicographically smaller than the other, say p, which means that p' is not canonical and hence would not have been output, a contradiction.

Conversely, let G be a cycle permutation graph of order n and $k := n/2$. We show that it gets output by Algorithm 7. Let p^k be the lexicographically minimal sequence representing G via any of its permutation 2-factors. Now suppose for $m < k$ that p^m is a canonical sequence such that $p^m = p^k|_m$. If we call Expand(p^m, m), it is clear that the algorithm checks whether $p^{m+1} := p^k|_{m+1}$ is canonical. We now show it is canonical and that the algorithm calls Expand(p^{m+1}, $m+1$) or terminates if p^{m+1} represent a cubic graph.

Assume $p := p^{m+1}$ is not canonical, then there is some sequence q representing the same graph via some 2-factor F' such that q is lexicographically smaller than p. In particular, this means that $p(i) = q(i) \neq ?$ for $i \in [l]$ for some $l < m + 1$, but $p(l) > q(l)$. This means however that q can be extended to a sequence q^k representing G via F' such that $q^k(i) = q(i)$ for all i for which $q(i) \neq ?$. From Proposition 5.3 it follows that every spoke in this extension is also a spoke of F and hence that F' is a permutation 2-factor of G. However, this is a contradiction, since this means that q^k is lexicographically smaller than p^k. Therefore, p is canonical. By induction, the graph represented by p^k, i.e. G will be output by the algorithm. □

The main bottleneck of this algorithm is dynamically updating the permutation 2-factors of the graph under consideration. In order to drastically speed up the computations, one could consider only checking one permutation 2-factor at the expense of generating some isomorphic copies for graphs having multiple permutation 2-factors, but of course when one is interested in the exact counts, this is only feasible when the number of output graphs is relatively small and post-processing can be done in reasonable time and using reasonable memory to filter the isomorphic copies and obtain the exact counts. In the case of non-hamiltonian cycle permutation graphs or permutation snarks the number of graphs generated was indeed relatively small, which is why we were able to apply the much faster approach that does not search for permutation 2-factors.

5.2.2 Non-hamiltonian cycle permutation graphs

Here we describe an adaptation of the previous algorithm in order to generate non-hamiltonian cycle permutation graphs. In Section 5.2.3 we do the same for permutation snarks. Next to extra pruning conditions that are specific to non-hamiltonian graphs and snarks (which will be described later in this section), the main difference is that we use the previously mentioned weaker form of canonicity, namely, by only considering the starting permutation 2-factor and not searching for any others. A rough outline for the recursive method of this approach (without any optimisations or technical details) is given by Algorithm 8. It is the same as Algorithm 7 but with the check *if p' is canonical* replaced by *if p' is weakly canonical*.

This approach, which we will denote as *weak orderly generation*, will generate all cycle permutation graphs for the given order, but will include several isomorphic copies in the output. However, since the counts of non-hamiltonian cycle permutation graphs and permutation snarks remain relatively low for small orders, it is feasible for these variations to perform an isomorphism check afterwards (e.g. using `nauty` [100]) in order to determine the actual counts.

Algorithm 8 Expand_Weak_Orderly(p, l)

if $l = k$ **then**
  Output cubic graph represented by p.
for all $x \in [k] \setminus \mathrm{im}(p)$ **do**
  Define p' by $p'(i) = p(i)$ for $i \in [k] \setminus \{l\}$ and $p'(l) = x$.
  if p' is weakly canonical **then**
    Expand_Weak_Orderly(p', $l + 1$)

A small adaptation to Algorithm 8 allows for the efficient generation of all non-hamiltonian cycle permutation graphs of a given order n. This is done by generating cycle permutation graphs of girth at least 5 (recall that a cycle of length 4 implies a hamiltonian cycle in these graphs) and by also rejecting an expansion if it leads to a hamiltonian graph. Note that because our algorithm works by adding edges, once a graph has a hamiltonian cycle, its descendants will also have one. We have adapted the program `cubhamg` included in `nauty` [100] to perform this hamiltonicity check as it is a very fast method for determining the hamiltonicity of (sub)cubic graphs. More details on this algorithm can be found in [20]. We also apply lookaheads to avoid performing the hamiltonicity check as much as possible, as for this algorithm the hamiltonicity check is the main bottleneck.

In particular, we forbid three types of hamiltonian cycles. Given two cycles C_1 and C_2 of a permutation 2-factor, a hamiltonian cycle has an even number of edges with an endpoint in C_1 and the other endpoint in C_2. We classify the hamiltonian cycles by how many of these edges they contain. Via lookaheads we forbid all hamiltonian cycles containing four and six of these edges. (Note that hamiltonian cycles containing two such edges are already dealt with, since they would introduce a cycle of length 4.)

For four such edges, one can look at the two previously added edges, and forbid a pair of edges from appearing consecutively. For example, suppose $C_1 = v_0 v_1 \dots v_{k-1}$ and $C_2 = w_0 w_1 \dots w_{k-1}$ are the two cycles of our permutation 2-factor under consideration. If $v_x w_{x'}$ and $v_y w_{y'}$ are the last two edges which were added, we can often forbid $v_z w_{z'}$ and $v_{z+1} w_{(z+1)'}$ from being added consecutively when $w_{z'}$ and $w_{(z+1)'}$ are neighbours of $w_{x'}$ and $w_{y'}$, respectively and vice versa. However, we cannot forbid these edges when they do not form a hamiltonian cycle, but instead form two disjoint cycles spanning the whole graph. Whether this is the case can easily be checked by analysing the order of $w_{x'}, w_{y'}, w_{z'}$ and $w_{(z+1)'}$ on C_2. A similar method can be used to forbid edges which will lead to hamiltonian cycles with six edges between C_1 and C_2.

The generation of non-hamiltonian cycle permutation graphs on orders 28, 30

Algorithm 9 Expand_Non-Hamiltonian(p, l)

```
if l = k then
    Output cubic graph represented by p.
for all x ∈ [k] \ im(p) do
    Define p' by p'(i) = p(i) for i ∈ [k] \ {l} and p'(l) = x.
    Let G be the graph represented by p'.
    if G has 4-cycle then
        return
    if G contains predetected hamiltonian cycle with 4 or 6 crossings then
        return
    if p' is weakly canonical and non-hamiltonian then
        Predetect future hamiltonian cycles with 4 or 6 crossings
        Expand_Non-Hamiltonian(p', l + 1)
```

and 32 is faster by a factor of 2 when using these lookaheads and seems to indicate a similar speedup for all orders. Our attempts to apply the same lookaheads for a general number of such edges or simply for hamiltonian cycles with eight edges between C_1 and C_2 did not lead to a further speedup of our program. A rough outline of the recursive method can be found in Algorithm 9.

5.2.3 Permutation snarks

We will now explain the variation which allows us to generate permutation snarks. The main idea is to filter out graphs containing a 2-factor in which all cycles have even length, i.e. *an even 2-factor*. If a cubic graph has such a 2-factor, it is 3-edge-colourable. We note that it is more efficient than generating non-hamiltonian cycle permutation graphs and filtering the snarks using post-processing (even though the latter is feasible for the orders we consider).

Similar ideas as in Section 5.2.2, with extensive pruning in the recursive method of the algorithm, lead to an efficient algorithm for the generation of permutation snarks, however empirical tests show that a heuristic approach, with less extensive pruning (and which might allow some graphs which are not permutation snarks to be output, but which in practice never happens up to order at least 46), is more efficient.

Again, we generate cycle permutation graphs of girth at least 5 (since a hamiltonian cycle is an even 2-factor) and reject an expansion if it introduces an even 2-factor in the graph. This is checked via backtracking in a way similar as the lookaheads from Section 5.2.2 for detecting hamiltonian cycles.

If the given graph does not have an even 2-factor, but the expansion with edge e does, then e must belong to the even 2-factor. We build 2-factors recursively. Let F be the current permutation 2-factor with cycles C_1 and C_2. We consider spokes $e_1 := e$ and e'_1 whose endpoint on C_2 is adjacent to the endpoint of e_1 on C_2. (Note that there might be two options for e'_1 and we recursively check both.) Then e_1 and e'_1 will belong to the 2-factor we are building and the edge on C_2 between the endpoints of e_1 and e'_1 will not. In the next step, we let e'_2 be a spoke such that its endpoint on C_1 is a neighbour of the endpoint of e_1 on C_1 and add a new pair e_2 and e'_2 to our 2-factor as before, where e_2 is a spoke whose endpoint on C_2 is adjacent to the endpoint on C_2 of e'_2. The edge on C_1 between the endpoints of e_1 and e'_2 and the edge on C_2 between e_2 and e'_2 will not belong to the 2-factor. We keep doing this recursively in all possible ways until the endpoint of e_i on C_1 is adjacent to the endpoint of e'_1 on C_1. Then we have obtained a 2-factor. If at any point we cannot find a spoke e_i or e'_i, the search backtracks.

Algorithm 10 Expand_Permutation_Snark(p, l)

```
if l = k then
    Output cubic graph represented by p.
for all x ∈ [k] \ im(p) do
    Define p' by p'(i) = p(i) for i ∈ [k] \ {l} and p'(l) = x.
    Let G be the graph represented by p'.
    if G has 4-cycle then
        return
    if G contains predetected hamiltonian cycle with 4 or 6 crossings then
        return
    if heuristic finds even 2-factor of G then
        return
    if p' is weakly canonical then
        Predetect future hamiltonian cycles with 4 or 6 crossings
        Expand_Permutation_Snark(p', l + 1)
```

Given a 2-factor and all of the spokes at which it crosses from C_1 to C_2, we can determine in time linear in this number of spokes, whether or not it is even. If it is even, we do not have a snark and can prune the current expansion. If not, we continue the search for even 2-factors. While this approach can find most even 2-factors relatively quickly, it is not exhaustive if we backtrack whenever we find a 2-factor. Whenever a 2-factor is encountered but not even, we can continue the search with a new pair e_i, e'_i, which is not necessarily adjacent to another spoke of our 2-factor, in all possible ways. However, empirical tests show that few extra even 2-factors are found at the expense of a lot of extra computational work. Therefore, we backtrack whenever we find a 2-factor. In

practice, up to order (at least) 46 no non-permutation snarks are output when running the algorithm using only this heuristic check for even 2-factors.

Finally, we combine this heuristic for finding even 2-factors with the predetection of hamiltonian cycles that cross four or six times which was used in Section 5.2.2. Although these would have been detected by the heuristic for even 2-factors, this small search often prunes faster than the depth-first heuristic. Adding this predetection speeds up the generation on orders 28–34 by a factor of approximately 1.7 and on order 36 by a factor of approximately 1.9. (Note that while no permutation snarks exist on order 0 mod 4, because any permutation 2-factor is even, we can still run through the search tree to get an idea of the algorithm's efficiency.) A rough outline of the recursive method for this approach can be found in Algorithm 10.

5.3 Results

In this section, we discuss the results of the computations that were performed using the implementations of the above mentioned algorithms. Our implementations are open source and can be found on GitHub [52]. Next to proving the algorithm's correctness (cf. Theorem 5.5), we also performed various tests for verifying the correctness of the implementation. These are described in Appendix D.2.

5.3.1 Cycle permutation graphs

We have used our algorithms to generate all cycle permutation graphs of a given order and also extended them to generate graphs with a lower bound on the girth. This can be done efficiently as the algorithm only adds edges, so we can prune as soon as we have a cycle which is smaller than the desired girth. The results obtained by our implementation of Algorithm 7 can be found in Table 5.1. These graphs can also be obtained from the House of Graphs [30] at `https://houseofgraphs.org/meta-directory/cubic` (up to the orders for which it was still feasible to store them). The runtimes which were needed to obtain these results can be found in Table D.1 in Appendix D.4.

While the implementation of the weak orderly generation approach (Algorithm 8) does not give exact counts, it is faster than Algorithm 7 at giving an upper bound for the number of pairwise non-isomorphic cycle permutation graphs for a given order. A comparison indicates that the factor with which the second algorithm is faster seems to grow as the order increases and the ratio

Order	$g \geq 4$	$g \geq 5$	$g \geq 6$	$g \geq 7$	$g \geq 8$	$g \geq 9$
8	2	0	0	0	0	0
10	4	1	0	0	0	0
12	10	1	0	0	0	0
14	28	3	0	0	0	0
16	123	11	1	0	0	0
18	667	59	0	0	0	0
20	4 815	402	4	0	0	0
22	41 369	3 602	9	0	0	0
24	411 231	37 178	84	0	0	0
26	4 535 796	424 252	846	1	0	0
28	54 828 142	5 289 603	12 597	0	0	0
30	717 967 102	71 206 645	197 921	1	0	0
32	10 118 035 593	1 027 074 710	3 334 149	6	0	0
34	152 626 831 184	15 800 380 281	58 638 599	190	0	0
36	?	258 386 596 744	1 077 159 843	4 437	1	0
38	?	?	20 642 970 164	147 820	0	0
40	?	?	?	5 166 381	0	0
42	?	?	?	167 517 630	2	0
44	?	?	?	5 265 419 873	33	0
46	?	?	?	?	847	0
48	?	?	?	?	21 294	0
50	?	?	?	?	1 053 289	0
52	?	?	?	?	7 281 578	0
54–58	?	?	?	?	?	0
60	?	?	?	?	?	2
62	?	?	?	?	?	61
64	?	?	?	?	?	1 654
66	?	?	?	?	?	71 213

Table 5.1: We omit the single cycle permutation graph of girth 3 on order 6 from this table. The columns with $g \geq k$ indicate the counts of cycle permutation graphs with girth at least k for each order.

of non-isomorphic graphs versus all graphs output by this algorithm seems to increase as well. For example on order 16, 96.85% of the output graphs are non-isomorphic, while on order 34, this is 99.21%. More details can be found in Table D.2 in Appendix D.4.

5.3.2 Non-hamiltonian cycle permutation graphs and permutation snarks

Certain adaptations to Algorithm 8 yield Algorithms 9 and 10, allowing for the efficient generation of all non-hamiltonian cycle permutation graphs or permutation snarks of a given order n, respectively.

Order	$g \geq 5$	$\chi' = 4$	$\lambda_c \geq 5$	$g \geq 6$	$g \geq 7$	$g \geq 8$	$g \geq 9$
10	1	1	1	0	0	0	0
12–16	0	0	0	0	0	0	0
18	2	2	0	0	0	0	0
20	0	0	0	0	0	0	0
22	1	0	0	0	0	0	0
24	0	0	0	0	0	0	0
26	64	64	0	0	0	0	0
28	0	0	0	0	0	0	0
30	9	0	0	0	0	0	0
32	0	0	0	0	0	0	0
34	10 778	10 771	12	0	0	0	0
36	4	0	0	0	0	0	0
38	1 848	0	0	0	0	0	0
40	19	0	0	0	0	0	0
42	3 131 740	3 128 893	736	0	0	0	0
44	1 428	0	0	0	0	0	0
46	678 106	0	0	0	0	0	0
48	?	?	?	0	0	0	0
50–54	?	?	?	?	0	0	0
56–60	?	?	?	?	?	0	0
62–70	?	?	?	?	?	?	0

Table 5.2: Counts of non-hamiltonian cycle permutation graphs for each order. Columns $g \geq k$ indicate counts with girth at least k for each order, the column $\chi' = 4$ indicates counts of permutation snarks and the column $\lambda_c \geq 5$ indicates counts of cyclically 5-edge-connected permutation snarks.

Using these adaptations we obtain the counts of non-hamiltonian cycle permutation graphs found in Table 5.2, and the following propositions, giving a partial answer to the questions asked by Klee [89] concerning the hamiltonicity of cycle permutation graphs. We also increase the previously known counts of permutation snarks from order 34 up to order 46. The graphs can be found in the meta-directory of the House of Graphs [30] at `https://houseofgraphs.org/meta-directory/snarks`. Note that trivially there exist no permutation snarks on order 48.

While, Algorithm 8 outputs isomorphic copies, we clarify that Table 5.2 contains the exact counts of non-hamiltonian cycle permutation graphs and permutation snarks as we filtered the isomorphic ones (using `nauty` [100]) whenever an adaptation of Algorithm 8 was applied. Since we also have an implementation of the canonical construction path method (see Algorithm 17 in Appendix D.1),

we were able to independently verify the counts up to order 42. Runtimes for the obtained counts using Algorithms 9 and 10 and Algorithm 17 can be found in Table D.3 and D.4, respectively, in Appendix D.5.

In contrast to the general case where the ratio of non-isomorphic graphs versus total graphs output by Algorithm 8 was close to 100% and increasing, in the non-hamiltonian case, the ratio is a lot lower and can vary a lot depending on the order. For example, during the generation of non-hamiltonian cycle permutation graphs on orders 34, 36 and 38, using Algorithm 9, we get 31.90%, 17.39% and 36.07%, respectively. With the generation of permutation snarks (on orders where they exist), using Algorithm 10, the ratio seems to decrease. We have on orders 26, 34 and 42, ratios 36.78%, 31.90% and 29.04%, respectively. (See Table D.5 in Appendix D.5 for more details.) An interesting observation is that non-hamiltonian cycle permutation graphs more often than not have multiple permutation 2-factors, which is opposite from the general case. Since isomorphisms occur when output graphs have multiple permutation 2-factors, this explains why there are many more isomorphic graphs output by the algorithm in this case.

It is easy to see that a permutation snark cannot have order 0 mod 4. Otherwise, the induced cycles of a permutation 2-factor are even and we can colour their edges using two colors and give all other edges the third color. Hence, a permutation snark can only exist for orders 2 mod 4. It was shown by Máčájova and Škoviera [96] that there exist (cyclically 5-edge-connected) permutation snarks on every order $n \equiv 2 \bmod 8$ for $n \geq 10$ ($n \geq 34$). However, so far no examples of order $n \equiv 6 \bmod 8$ are known and in fact this was posed as an open problem in [18, Problem 1], where all permutation snarks up to order 34 were determined. Máčájova and Škoviera proved [96] that a smallest example, if it exists, must be cyclically 5-edge-connected. From the permutation snarks, we filtered the ones which are cyclically 5-edge-connected. These counts can be found in the fourth column of Table 5.2.

Using the results of Algorithm 10 found in Table 5.2, we showed the following.

Proposition 5.6. *A smallest permutation snark of order* 6 mod 8 *has order at least* 54*.*

We also obtained this in a second way, namely using Algorithm 9, we generated all non-hamiltonian cycle permutation graphs up to order 46. See Table 5.2. While there are 1 848 non-hamiltonian cycle permutation graphs of order 38 and 879 828 such graphs of order 46, using two independent algorithms, we verified that all of them are 3-edge-colourable.

Zhang's conjecture [148] stating that the Petersen graph is the only cyclically 5-edge-connected permutation snark was refuted by Brinkmann et al. [18]. Our

search verified their 12 counterexamples on order 34 and found 736 new ones on order 42, cf. Table 5.2.

While Shawe-Taylor and Pisanski showed in [109] that the girth of cycle permutation graphs can be arbitrarily large, no permutation snarks of girth 6 or higher are known. Our computational results from Table 5.2 imply the following.

Proposition 5.7. *The smallest non-hamiltonian cycle permutation graph of girth at least* 6 *has order at least* 50*. The smallest such graph of girth at least* 7 *has order at least* 56*. The smallest such graph of girth at least* 8 *has order at least* 62*. The smallest such graph of girth at least* 9 *has order at least* 72*.*

Note that as a corollary, we can replace "non-hamiltonian cycle permutation graph" in the above proposition with "permutation snark".

5.4 Orders of non-hamiltonian cycle permutation graphs

In his 1972 paper [89], Klee studied two questions involving the non-hamiltonicity of cycle permutation graphs, attributed to Ralph Willoughby. The first one asks which permutation graphs admit a hamiltonian cycle. The second asks for which orders n there exist non-hamiltonian cycle permutation graphs. Klee answered this second question partially by proving that for all $n \equiv 2 \bmod 4$ there is a non-hamiltonian cycle permutation graph if and only if n is neither 6 nor 14. We complete the characterisation of the orders for which non-hamiltonian cycle permutation graphs exist and thereby solve Klee's second question.

We use Klee's notation, but opt for 0-indexing of the permutations. One can define cycle permutation graphs of order $n = 2k$ given a k-permutation. Let $G(\pi)$ be the cubic graph consisting of two disjoint k-cycles $x_0x_1 \dots x_{k-1}$ and $y_0y_1 \dots y_{k-1}$ and edges $x_iy_{\pi(i)}$ for $0 \leq i \leq k-1$. Let $G'(\pi)$ be the subgraph of $G(\pi)$ where we have removed the edges $x_{k-1}x_0$ and $y_{k-1}y_0$.

An E-path in the graph $G'(\pi)$ is a path for which its ends are both in the set $E = \{x_0, x_{k-1}, y_0, y_{k-1}\}$. A hamiltonian E-pair is a pair of disjoint E-paths containing all vertices of $G'(\pi)$. The permutation π is called *good* if it either has a hamiltonian E-path with endpoints x_i and y_j for $i, j \in \{0, k-1\}$ or if it has a hamiltonian E-pair such that one of the disjoint paths has endpoints x_i and y_j for $i, j \in \{0, k-1\}$ and the other has endpoints x_{k-1-i}, y_{k-1-j}. Otherwise, π is *bad*.

If π is a k-permutation with $\pi(k-1) = k-1$, then we can naturally restrict π from $[k]$ to $[k-1]$. We denote this permutation by $\overline{\pi}$, i.e. $\overline{\pi}(i) = \pi(i)$ for $i \in [k-1]$.

Given m k_i-permutations, for $0 \leq i < m$, we can concatenate them to obtain a new $(\sum_{i=0}^{m-1} k_i)$-permutation $\pi := (\pi_0, \ldots, \pi_{m-1})$, where

$$\pi(i) = \sum_{j=0}^{l-1} k_j + \pi_i(i - \sum_{j=0}^{l-1} k_j), \text{ for } \sum_{j=0}^{l-1} k_j \leq i < \sum_{j=0}^{l} k_j.$$

We will use the following propositions.

Proposition 5.8 (Klee [89])**.** *If π is a k-permutation with $\pi(k) = k \geq 3$, then $G(\pi)$ is non-hamiltonian if and only if the $(k-1)$-permutation $\overline{\pi}$ is bad.*

Proposition 5.9 (Klee [89])**.** *Suppose that π_i is a k_i permutation for $0 \leq i < m$ and that the following three conditions hold:*

1. *for each i, either $k_i = 1$ or π_i is bad;*
2. *there is an even (possibly zero) number of i's for which $k_i = 1$;*
3. *$k_0 \neq 1$, $k_{m-1} \neq 1$ and if $k_i = k_j = 1$ we have that $j \neq i+1$.*

Then the concatenation $\pi = (\pi_0, \ldots, \pi_{m-1})$ is bad.

We can now finish the characterisation of the orders.

Proposition 5.10. *There is a non-hamiltonian cycle permutation graph of order n if and only if n is even and $n \in \{10, 18, 22, 26, 30\}$ or $n \geq 34$.*

Proof. The orders $n \equiv 2 \bmod 4$ were already characterised by Klee. By the counts of Table 5.2 we see that all cycle permutation graphs of order $n \equiv 0 \bmod 4$ and $n \leq 32$ are hamiltonian.

Let π_P be a bad 4-permutation of the Petersen graph (the non-hamiltonian cycle permutation graph on order 10), which exists due to Proposition 5.8. Let G_1 be a non-hamiltonian cycle permutation graph on order 36 and $\pi_{G_{36}}$ a bad 17-permutation obtained from this graph, let G_2 be a cycle permutation graph on order 40 and $\pi_{G_{40}}$ be a bad 19-permutation of this graph and let π_1 be the trivial 1-permutation. Such a G_1 and G_2 exist, see Table 5.2.

By Proposition 5.9, we obtain that all permutations in the following sequence are bad.

$$\pi_{G_{36}}, \pi_{G_{40}}, (\pi_{G_{36}}, \pi_P), (\pi_{G_{40}}, \pi_P),$$
$$(\pi_{G_{36}}, \underbrace{\pi_P, \ldots, \pi_P}_{k \text{ times}}), (\pi_{G_{36}}, \pi_1, \pi_P, \pi_1, \underbrace{\pi_P, \ldots, \pi_P}_{l \geq 1 \text{ times}})$$

By Proposition 5.8, these bad permutations imply the existence of a non-hamiltonian cycle permutation graph on order, respectively, $36, 40, 44, 48, 36 + 8k, 48 + 8l$, where $l \geq 1$, which finishes the proof. □

Acknowledgements

The authors thank Gunnar Brinkmann, Edita Máčajová and Martin Škoviera for their valuable insights and contributions.

This research of Jan Goedgebeur and Jarne Renders was supported by Internal Funds of KU Leuven and by an FWO grant with grant number G0AGX24N. Several of the computations for this work were carried out using the supercomputer infrastructure provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government.

Part III

Acyclic Graphs

Chapter 6

HIST-Critical Graphs and Malkevitch's Conjecture

This chapter is a minimally edited version of the following submitted work.

J. Goedgebeur, K. Noguchi, J. Renders and C. T. Zamfirescu. HIST-critical graphs and Malkevitch's conjecture. *arXiv preprint*, 2024. arXiv: `2401.04554`

My contributions to this paper, in accordance with the contributor role taxonomy (CRediT)[1], consist of: Conceptualisation, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review and Editing, and Visualisation.

6.1 Introduction

In a given graph, a spanning tree without 2-valent vertices is called a *HIST*, an abbreviation of *homeomorphically irreducible spanning tree*. A graph not containing a HIST is *HIST-free*. HIST-free graphs play an important role in the theory of these spanning trees, see for instance the work of Albertson, Berman, Hutchinson, and Thomassen [1], and many fundamental questions remain unanswered. We will call a graph G K_1*-histonian* if every vertex-deleted subgraph of G contains a HIST. In this article our aim is to investigate HIST-freeness from two perspectives: in the first part we focus on *HIST-critical* graphs, i.e. HIST-free K_1-histonian graphs, e.g. K_3. In the second part we

[1] See `https://credit.niso.org/`.

study Malkevitch's Conjecture stating that planar 4-connected graphs must contain a HIST. We point out that the question whether a graph contains a HIST or not has been intensely investigated, see for instance [26, 27, 76, 103].

We recall that a cycle in a graph is *hamiltonian* if it visits every vertex of the graph, and a graph is *hamiltonian* if it contains a hamiltonian cycle. So in a given graph a hamiltonian cycle is a connected spanning subgraph in which *every* vertex has degree 2, while a HIST is a connected spanning subgraph in which *no* vertex has degree 2. Just like the conjecture of Malkevitch [97] stating that every planar 4-connected graph contains a HIST aims at establishing a HIST-analogue of Tutte's celebrated theorem that planar 4-connected graphs are hamiltonian [133], much of the work here is motivated by the desire to better understand in which cases hamiltonian cycles and hamiltonicity-related concepts behave like their HIST counterparts, and in which cases they do not (and, of course, why this is so). We remark that the notion "HIST-critical" has a hamiltonian counterpart in "*hypohamiltonian*": these are non-hamiltonian graphs in which every vertex-deleted subgraph is hamiltonian.

Throughout the article we will make use of a combination of theoretical and computational arguments. Therefore, in Section 6.2 we first present the algorithm we used to test whether or not a graph is HIST-free and to count its number of HISTs if it is not. In Section 6.3 we focus on HIST-critical graphs and give an almost complete characterisation of the orders for which these graphs exist and present an infinite family of planar examples which are 3-connected and in which nearly all vertices are 4-valent. In Section 6.4 we prove a series of results motivated by Malkevitch's Conjecture. We show by computational means that it holds up to at least 22 vertices, determine the minimum number of HISTs in planar 4-connected graphs, prove that antiprisms of order $2k$ with $k \geq 3$ have exactly $2k(2k-2)$ HISTs, and a short proof of the result that there exists a 4-connected HIST-free graphs of genus at most 39.

We shall use the notation $[n] := \{0, \ldots, n\}$. For a graph G and a subgraph H of G, we write the number of neighbours of v in H as $d_H(v)$ and put $d(v) := d_G(v)$. We call a vertex of degree k a *k-vertex*.

6.2 Algorithm for counting HISTs

We implemented an efficient branch and bound algorithm to test whether or not a graph is HIST-free and to count its number of HISTs if any are present. The main idea of our algorithm is a straightforward way of searching for the spanning trees of the graph G by recursively adding edges to a tree T and

forbidding edges from being added. These forbidden edges induce a subgraph of G, say G'.

Given some subtree T and subgraph G' of our graph, we find a vertex v for which $d_G(v) - d_{G'}(v) - d_T(v)$ is non-zero but minimal and which contains a neighbour w such that $vw \notin E(G')$ and $vw \notin E(T)$. Let w be this neighbour for which $d_G(w) - d_{G'}(w) - d_T(w)$ is minimal. At this point we branch. We add vw to G', forbidding it from being added to the tree in this branch, and recurse. If w does not already belong to T, we add vw to T and recurse.

A spanning tree is found if $|E(T)| = |V(G)| - 1$ and it is a HIST if there are no vertices of degree 2. We use certain additional elementary pruning criteria. For example, if for a vertex v we have $d_G(v) = d_{G'}(v)$ or $d_G(v) - d_{G'}(v) = d_T(v) = 2$, we can prune the current branch.

Stopping the search once a HIST is found gives us an algorithm for checking whether a graph is HIST-free. This algorithm can then also be used to determine whether a graph is K_1-histonian or HIST-critical.

While the correctness of the algorithm can easily be proven, there is always the risk of errors in the implementation of the algorithm. To mitigate these we verified many of our results using an independent implementation of this algorithm as well as the implementation of a different branch and bound algorithm, which starts from an initial spanning tree and alters it by adding an removing edges in such a way that we will have encountered all HISTs, similar to the algorithm of Kapoor and Ramesh [86] for spanning trees. For details on the verification, we refer the reader to Appendix E.1.

Our implementation of the algorithm is open source software and can be found on GitHub [50] where it can be verified and used by other researchers.

6.3 HIST-Critical Graphs

In the theory of hamiltonicity, so-called *hypohamiltonian* graphs—these are non-hamiltonian graphs in which every vertex-deleted subgraph is hamiltonian—play a special role. The smallest such graph is the famous Petersen graph. The investigation of this class of graphs started in the sixties and results have appeared in a steady stream throughout the decades. In the beginning, it seemed that they were closely related to snarks—many snarks being hypohamiltonian graphs and vice-versa—but as more and more examples were described, it became clear that the families are quite different. In line with this early perceived similarity, Chvátal [28] asked whether *planar* hypohamiltonian graphs exist, and Grünbaum [73] conjectured their non-existence; recall that the non-

planarity of snarks is equivalent to the Four Colour Theorem. Thomassen proved, using Grinberg's hamiltonicity criterion as an essential tool, that infinitely many planar hypohamiltonian graphs exist [130].

In this section, we look at the HIST-analogue of hypohamiltonian graphs, namely HIST-critical graphs. As mentioned in the introduction, these are HIST-free graphs in which every vertex-deleted subgraph does contain a HIST. We mirror the development in hypohamiltonicity theory in two ways: first we give a near-complete characterisation of orders for which HIST-critical graphs exist (this was completed for hypohamiltonian graphs by Aldred, McKay, and Wormald [2]), and then show that infinitely many planar HIST-critical graphs exist, paralleling Thomassen's aforementioned result.

The latter family consists of 3-connected graphs in which nearly all vertices are 4-valent. Our desire to find planar HIST-critical graphs with few cubic vertices is motivated as follows. Thomassen also proved the surprising structural result that every planar hypohamiltonian graph must contain a cubic vertex [128]. Note that this is equivalent to the statement that every planar graph with minimum degree at least 4 and in which every vertex-deleted subgraph is hamiltonian must be itself hamiltonian—this strengthens Tutte's celebrated theorem that planar 4-connected graphs are hamiltonian. It remains unknown whether every planar HIST-critical graph must contain a cubic vertex.

Clearly, K_3 is the smallest HIST-critical graph. The first question one can ask, in the spirit of establishing which parallels between HIST-critical and hypohamiltonian graphs hold and which do not, is whether there is a HIST-analogue of the Petersen graph. One key property of the Petersen graph is that it is 3-regular. We now give the easy proof of a fact that makes it impossible to find a suitable HIST-analogue of the Petersen graph.

Proposition 6.1. *In a graph of even order and maximum degree at most* 3*, there exists no vertex-deleted subgraph with a HIST. In particular, there are no* 3*-regular HIST-critical graphs.*

Proof. Any HIST T of a graph H of maximum degree at most 3 contains only 1- and 3-vertices. The difference between the number of 1- and 3-vertices present in T must be 2. So the order of T and thus of H must be even. But the graph G from the statement is required to have even order, so its vertex-deleted subgraphs must have odd order. □

In order to obtain other examples—in particular, in light of the above observation, examples of *even* order—, we used `geng` [100] to exhaustively generate general 2-connected graphs and used our algorithm from Section 6.2 to test which of the generated graphs are HIST-critical. Note that HIST-critical

graphs are 2-connected. The results are summarised in Table 6.1. The table shows the existence of several HIST-critical graphs other than K_3, but none of even order. Illustrations of the five smallest HIST-critical graphs can be found in the appendix, see Figure E.1. In the hope of finding more examples, we also computed HIST-critical graphs under girth restrictions, as this allowed us to look at higher orders. A HIST-critical graph with girth $4, 5, 6$ and 7 can also be found in the appendix, see Figure E.2. The results can also be found in Table 6.1. All graphs from this table can be downloaded from the *House of Graphs* [30] at `https://houseofgraphs.org/meta-directory/hist-critical`. Now we did find examples of even order. We shall now prove that there are in fact infinitely many such graphs.

Order	$h(n)$	$h(4,n)$	$h(5,n)$	$h(6,n)$	$h(7,n)$
3	1	0	0	0	0
4, 5, 6	0	0	0	0	0
7	2	0	0	0	0
8	0	0	0	0	0
9	2	0	0	0	0
10	0	0	0	0	0
11	35	3	1	0	0
12	0	0	0	0	0
13	153	6	2	0	0
14	?	1	1	0	0
15	?	149	25	0	0
16	?	3	0	0	0
17	?	?	244	0	0
18	?	?	1	0	0
19	?	?	4 129	4	0
20	?	?	3	1	0
21	?	?	?	98	0
22	?	?	?	0	0
23	?	?	?	6 036	0
24	?	?	?	52	0
25, 26	?	?	?	?	0
27	?	?	?	?	8

Table 6.1: Exact counts of HIST-critical graphs with a given lower bound on the girth. Column $h(k,n)$ gives the number of n-vertex HIST-critical graphs with girth at least k. We put $h(n) := h(3,n)$.

During our search, we noticed that, given a cubic graph (with girth restrictions), one sometimes obtains a HIST-critical graph by subdividing the proper edges.

See for example the graph of Figure E.3 in the appendix. It is a HIST-critical graph of girth 6 obtained by subdividing the Pappus graph in three places.

Using this observation and starting from cubic graphs of girth equal to 8 and 9, we were able to find HIST-critical graphs of girth 8 of order $37, 39$ and 41 and of girth 9 of order 59 in a non-exhaustive way. An example of such a girth 8 graph of order 41 can be found at `https://houseofgraphs.org/graphs/50549` and an example of such a girth 9 graph of order 59 can be found at `https://houseofgraphs.org/graphs/50547`. Their existence will be used in the proof of Theorem 6.6.

6.3.1 HIST-Critical fragments

Ultimately, our goal is a HIST-analogue of the result of Aldred, McKay, and Wormald [2] stating that there exists a hypohamiltonian graph of order n if and only if $n \in \{10, 13, 15, 16\}$ or $n \geq 18$. Although our characterisation, given in Section 6.3.4, is not complete, only few orders remain open.

Let F be a graph with $V(F) = \{x, y, v_1, \ldots, v_\ell\}$ where $\ell \geq 1$. For a given connected subgraph H of F, we call a spanning tree (spanning forest) Υ of H an *$\{x, y\}$-excluded HIST* (*$\{x, y\}$-excluded HISF)* if $d_\Upsilon(v) \neq 2$ for all $v \in V(H) \backslash \{x, y\}$. A graph F will be called a *HIST-critical $\{x, y\}$-fragment* if it satisfies all of the following properties.

1. F has an $\{x, y\}$-excluded HIST. Moreover, every $\{x, y\}$-excluded HIST T of F satisfies $d_T(x) = d_T(y) = 2$;
2. $F - x$ has an $\{x, y\}$-excluded HIST with $d_F(y) \neq 1$ and $F - y$ has an $\{x, y\}$-excluded HIST with $d_F(x) \neq 1$;
3. for every $v \in V(F) \backslash \{x, y\}$, the graph $F - v$ either has (a) an $\{x, y\}$-excluded HIST with at least one of x and y of degree $\neq 2$, or (b) an $\{x, y\}$-excluded HISF consisting of exactly two components T_x and T_y, each on at least two vertices, such that $x \in V(T_x)$ and $y \in V(T_y)$; and
4. F does not have an $\{x, y\}$-excluded HISF with property (3b) above.

Theorem 6.2. *Let $k \geq 2$ be an integer and denote by C two vertices connected by two parallel edges if $k = 2$ and the k-cycle $v_0 \ldots v_{k-1} v_0$ if $k \geq 3$. For all $i \in [k - 1]$, consider pairwise disjoint HIST-critical $\{x_i, y_i\}$-fragments H_i, all disjoint from C. For all i, identify v_i with x_i and v_{i+1} with y_i, and remove the edge $v_i v_{i+1}$, indices mod. k. The resulting graph G is HIST-critical.*

Proof. In this proof we see H_i as a subgraph of G for every $i \in [k-1]$. We first show that G is K_1-histonian. By (1), for every $i \in [k-1]$ the graph H_i contains an $\{x_i, y_i\}$-excluded HIST T_i satisfying $d_{T_i}(x_i) = d_{T_i}(y_i) = 2$. By (2), $H_0 - x_0$ has an $\{x_0, y_0\}$-excluded HIST T_0' with $d_{T_0}(y_0) \neq 1$ and $H_{k-1} - y_{k-1}$ has an $\{x_{k-1}, y_{k-1}\}$-excluded HIST T_{k-1}' with $d_{T_{k-1}}(x_{k-1}) \neq 1$. Then, since $x_0 = y_{k-1}$, the tree $T_0' \cup \bigcup_{i=1}^{k-2} T_i \cup T_{k-1}'$ is a HIST of $G - x_0$. Finding a HIST of $G - v$ is analogous for any other vertex $v \in \{x_i, y_i\}_{i \in [k-1]}$.

Consider $v \in V(H_0) \setminus \{x_0, y_0\}$. By (3), $H_0 - v$ either has (a) an $\{x_0, y_0\}$-excluded HIST S with at least one of x_0 and y_0 of degree $\neq 2$, or (b) an $\{x_0, y_0\}$-excluded HISF consisting of exactly two components S_{x_0} and S_{y_0}, each on at least two vertices, such that $x_0 \in V(S_{x_0})$ and $y_0 \in V(S_{y_0})$. We first treat case (a). We may assume without loss of generality $d_S(x_0) \neq 2$. Then $S \cup \bigcup_{i=1}^{k-2} T_i \cup T_{k-1}'$ is a HIST of $G - v$, where $T_1, \ldots, T_{k-2}, T_{k-1}'$ are defined as in the preceding paragraph. For case (b), the tree $S_{y_0} \cup \bigcup_{i=1}^{k-1} T_i \cup S_{x_0}$ is a HIST of $G - v$.

We now show that G is HIST-free. Assume G does contain a HIST T. Put $T_i := T \cap H_i$. A *HISF* shall be a disjoint union of HISTs. By construction and in particular by (1) (we should rule out why $\bigcup_{i=0}^{k-2} T_i \cup T_{k-1}'$ is not a HIST) there exists a $j \in [k-1]$ such that T_i is a HIST for all $i \in [k-1] \setminus \{j\}$ and T_j is a HISF consisting of exactly two components, one containing x_j, the other containing y_j. A priori, one of these components might be isomorphic to K_1, but this is in fact impossible: every HIST of T_i is an $\{x_i, y_i\}$-excluded HIST of T_i, so by (1) the T_i-degrees of x_i and y_i must be 2, so single-vertex components in the aforementioned HISF cannot occur because this would signify the presence of a 2-vertex in T. Every HISF of T_j is also an $\{x_j, y_j\}$-excluded HISF of T_j. But now the existence of T_j contradicts (4), so G is HIST-free. □

First, we remark that the degree requirements on x and y in property (3) are only necessary when $k = 2$.

Let F_1 and F_2 be graphs defined as follows; see also Figures E.4 and E.5 in the Appendix. Note that F_1 is the Petersen graph from which two adjacent vertices were removed.

$$
\begin{aligned}
V(F_1) &= \{x, y, v_1, \ldots, v_6\} \\
E(F_1) &= \{xv_3, xv_6, yv_1, yv_4, v_1v_2, v_1v_6, v_2v_3, v_3v_4, v_4v_5, v_5v_6\} \\
V(F_2) &= \{x, y, v_1, \ldots, v_{10}\} \\
E(F_2) &= \{xv_1, xv_8, yv_6, yv_9, v_1v_2, v_1v_6, v_1v_7, v_2v_3, v_3v_4, v_3v_8, \\
&\qquad v_4v_5, v_4v_9, v_5v_6, v_6v_{10}, v_7v_9, v_8v_{10}\}
\end{aligned}
$$

Proposition 6.3. *The graphs F_1 and F_2 are HIST-critical $\{x, y\}$-fragments.*

Proof. For F_1 and F_2, properties (2) and (3) can be checked by Figures E.4 and E.5 in the Appendix, using symmetry. Now we confirm properties (1) and (4).

For F_1, let Υ be either an $\{x, y\}$-excluded HIST or an $\{x, y\}$-excluded HISF with property (3b). In Υ, precisely one of the two edges v_1v_2 or v_2v_3 are used. If $v_1v_2 \in E(\Upsilon)$, then $yv_1, v_1v_6 \in E(\Upsilon)$. Similarly, precisely one of v_4v_5 or v_5v_6 is present. On the one hand, if $v_5v_6 \in E(\Upsilon)$, then $xv_6, v_1v_6 \in E(\Upsilon)$ and we see that Υ is the $\{x, y\}$-excluded HIST with

$$E(\Upsilon) = \{xv_3, xv_6, yv_1, yv_4, v_1v_2, v_1v_6, v_5v_6\}.$$

(See the top centre drawing of Figure E.4.) On the other hand, if $v_4v_5 \in E(\Upsilon)$, then $yv_4, v_3v_4 \in E(\Upsilon)$ and, hence, x cannot be in Υ which is a contradiction. If the edge $v_2v_3 \in E(\Upsilon)$, then the same argument implies that Υ is the $\{x, y\}$-excluded HIST with

$$E(\Upsilon) = \{xv_3, xv_6, yv_1, yv_4, v_2v_3, v_3v_4, v_4v_5\}.$$

In both cases, Υ is an $\{x, y\}$-excluded HIST with $d_\Upsilon(x) = d_\Upsilon(y) = 2$.

For F_2, let Υ be either an $\{x, y\}$-excluded HIST or an $\{x, y\}$-excluded HISF with the property (3b). In Υ, precisely one of the two edges v_1v_2 or v_2v_3 are used and precisely one of the two edges v_4v_5 or v_5v_6 are used. We consider the three cases by symmetry: $v_1v_2, v_4v_5 \in E(\Upsilon)$, $v_1v_2, v_5v_6 \in E(\Upsilon)$, and $v_2v_3, v_4v_5 \in E(\Upsilon)$. If $v_1v_2, v_4v_5 \in E(\Upsilon)$, then $v_3v_4, v_4v_9 \in E(\Upsilon)$ and, hence, $yv_9, v_7v_9 \in E(\Upsilon)$. To include the two vertices v_8 and v_{10}, we see that Υ is the $\{x, y\}$-excluded HIST with

$$E(\Upsilon) = \{xv_1, xv_8, yv_6, yv_9, v_1v_2, v_1v_6, v_3v_4, v_4v_5, v_4v_9, v_6v_{10}, v_7v_9\}.$$

If $v_1v_2, v_5v_6 \in E(\Upsilon)$, then $v_3v_8, v_4v_9 \in E(\Upsilon)$ to include the two vertices v_3 and v_4 and, hence, $xv_8, yv_9, v_7v_9, v_8v_{10} \in E(\Upsilon)$. Then we see that Υ is the $\{x, y\}$-excluded HIST with

$$E(\Upsilon) = \{xv_1, xv_8, yv_6, yv_9, v_1v_2, v_1v_6, v_3v_8, v_4v_9, v_5v_6, v_7v_9, v_8v_{10}\}.$$

(See the top centre drawing of Figure E.5.) If $v_2v_3, v_4v_5 \in E(\Upsilon)$, then $v_3v_4, v_3v_8, v_4v_9 \in E(\Upsilon)$ and, hence, by symmetry we can assume that $xv_8, v_8v_{10} \in E(\Upsilon)$. If $yv_9 \notin E(\Upsilon)$, then $yv_6 \in E(\Upsilon)$ and we see that v_7 cannot be in Υ. So we have $yv_9 \in E(\Upsilon)$ and we see that Υ is the $\{x, y\}$-excluded HIST with $E(\Upsilon) = \{xv_1, xv_8, yv_6, yv_9, v_2v_3, v_3v_4, v_3v_8, v_4v_5, v_4v_9, v_7v_9, v_8v_{10}\}$. In all cases, Υ is an $\{x, y\}$-excluded HIST with $d_\Upsilon(x) = d_\Upsilon(y) = 2$. □

6.3.2 Gluing K_1-histonian graphs

One might wonder whether gluing procedures – which are very successful in the context of hypohamiltonian graphs – can be formulated for HIST-critical graphs. Unfortunately, the next observation shows that a very natural gluing procedure applied to two K_1-histonian graphs (this includes all HIST-critical graphs) always yields a graph containing a HIST.

Let G and H be disjoint graphs. We consider non-adjacent vertices x_G, y_G in G and non-adjacent vertices x_H, y_H in H. First, identify x_G with x_H and y_G with y_H; the obtained vertices will be denoted by x and y. Thereafter, add a new vertex z and join it to x and y. Finally, add the edge xy. The resulting graph shall be denoted by $(G, x_G, y_G) : (H, x_H, y_H)$, and when the choice of x_G, y_G and x_H, y_H plays no role, we simply write $G : H$. Observe that this can be seen as identifying two non-adjacent vertices in two K_1-histonian graphs, and then identifying these two vertices with two vertices of a triangle, which is HIST-critical.

Proposition 6.4. If G and H are K_1-histonian, then $G : H$ is K_1-histonian. Moreover, $G : H$ contains a HIST.

Proof. Throughout the proof we see G and H as subgraphs of $\Gamma := G : H$. In $\Gamma - x$, we obtain a HIST T by taking the union of the HIST present in $G - x$, the HIST present in $H - x$, and $(\{y, z\}, yz)$, thus guaranteeing that the degree of y in T is at least 3. Analogously we obtain a HIST in $\Gamma - y$. In $\Gamma - z$, consider $T - yz + xy$. By considering $T + xy$, we see that Γ itself must contain a HIST.

Now let v be a vertex in $G - x - y$. Consider a HIST T_G^v in $G - v$ and a HIST T_H^x in $H - x$. Then $T_G^v \cup T_H^x \cup (\{y, z\}, yz)$ is the desired HIST in $\Gamma - v$. For a vertex in $H - x - y$ we can use the same argument, thus completing the proof. □

6.3.3 Planar HIST-critical graphs

Here we give an exhaustive list of the counts of all planar HIST-critical graphs up to order 14, and present an infinite family of planar HIST-critical graphs.

Using `plantri` [21] we generated all planar 2-connected graphs up to order 14 and used our algorithm from Section 6.2 to determine which are HIST-critical. The results can be found in Table 6.2. All graphs from this table can be obtained from the *House of Graphs* [30] at `https://houseofgraphs.org/meta-directory/hist-critical` and also be inspected in the database of interesting

graphs at the House of Graphs by searching for the keywords "planar HIST-critical".

Order	3	4	5	6	7	8	9	10	11	12	13	14
HIST-critical	1	0	0	0	2	0	0	0	12	0	12	0

Table 6.2: Exact counts of planar HIST-critical graphs for each order.

Motivated by corresponding problems for hypohamiltonian graphs (as described at the beginning of this section), we shall now present an infinite family of HIST-critical graphs, with the added property of planarity. The second part of the next theorem is motivated by another parallel to hypohamiltonicity: Chvátal conjectured in [28] that if the deletion of an edge e from a hypohamiltonian graph G does not create a vertex of degree 2, then $G - e$ is hypohamiltonian. Thomassen [129] gave numerous counterexamples to the aforementioned conjecture, yet none of them were planar, and the last author gave planar counterexamples [144]. We now show that the same is true for HIST-critical graphs.

Theorem 6.5. There exist infinitely many planar HIST-critical graphs. Moreover, there exist infinitely many planar HIST-critical graphs G, each containing an edge e such that $G - e$ is 3-connected and HIST-critical.

Proof. For each integer $k \geq 3$, let G_k be a planar graph with vertex set and edge set defined as follows.

$$\begin{aligned} V(G_k) &= \{a_1, \ldots, a_k, b_1, \ldots, b_k, c_1, \ldots, c_{k-1}, x, y\} \\ E(G_k) &= \{a_i a_{i+1}, a_i c_i, a_{i+1} c_i, b_i b_{i+1}, b_i c_i, b_{i+1} c_i \mid 1 \leq i \leq k-1\} \cup \\ &\quad \{a_1 x, a_k x, b_1 y, b_k y, xy\} \end{aligned}$$

Its plane embedding is depicted in Figure 6.1a, where x and y are adjacent. (It is not difficult to see that G_k is 3-connected, and this embedding is unique by a classic result of Whitney.)

We show that, for every even integer $k \geq 4$, both the graph G_k and the graph $H_k := G_k + a_1 a_k$ are planar HIST-critical graphs. Thus, H_k is the desired infinite family. First, we prove that H_k is HIST-free. Suppose H_k does have a HIST T. Since $|V(T)| = 3k + 1$ is odd, H has a vertex of even degree by the handshaking lemma, that is, a 4-vertex v. Note that every 4-vertex in H_k lies on two adjacent triangles vpq and vrs. Thus, exactly four edges, namely vp, vq, vr, vs in the two triangles lie in T. Since T spans all vertices of H_k, at

least one of p, q, r, s should be of degree 3 in T, say, p. Then p should have degree 4 in H_k and p lies on another triangle, say, ptu, and so exactly two edges pt, pu in the triangle must be in T. By this argument, it is easy to see that T contains none of the three edges b_1y, b_ky, xy which do not lie on a triangle, and hence T does not contain the vertex y, a contradiction. Hence, G_k is HIST-free, too.

Next, we show that for every $v \in V(G_k)$, $G_k - v$ has a HIST. By symmetry, we only need to consider the following five cases.

- For $v = x$, see Figure 6.1b.
- For $v = a_i$ where $i \in \{1, 3, \ldots, k-3\}$, see Figure 6.1c. One should add edges

$$a_1c_1, a_2c_1, a_3c_3, a_4c_3, \ldots, a_{i-2}c_{i-2}, a_{i-1}c_{i-2}, a_{i+1}c_{i+1}, a_{i+2}c_{i+1}, \ldots,$$

$$a_{k-4}c_{k-4}, a_{k-3}c_{k-4}.$$

- For $v = a_{k-1}$, see Figure 6.1d.
- For $v \in \{c_1, c_3, \ldots, c_{k-1}\}$, see Figure 6.1e.
- For $v \in \{c_2, c_4, \ldots, c_{k-2}\}$, see Figure 6.1f.

It follows that for every $v \in V(H_k)$, H_k has a HIST, too. □

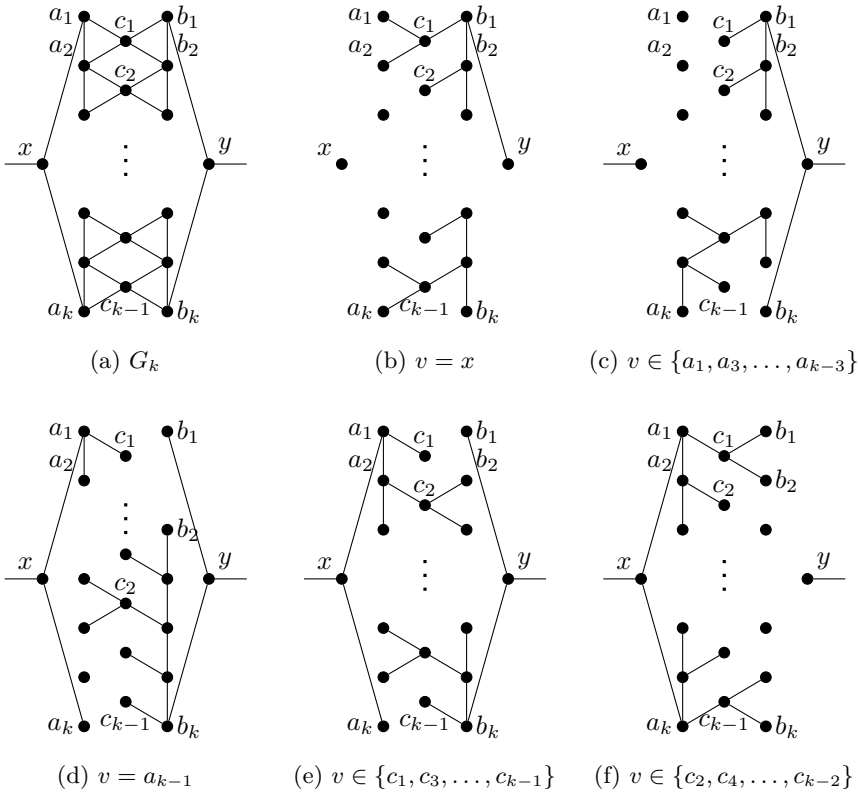
6.3.4 A near characterisation of the orders for which HIST-critical graphs exist

We summarise our computations and theoretical arguments regarding the existence and non-existence of HIST-critical graphs in the following result.

Theorem 6.6. *Let*

$$\mathcal{N} := \{1, 2, 4, 5, 6, 8, 10, 12\}, \qquad \mathcal{M} := \{26, 30, 34, 38, 45, 48, 52\}.$$

There exist HIST-critical graphs for every $n \in \mathbb{N} \setminus (\mathcal{N} \cup \mathcal{M})$, while there are no HIST-critical graphs of order $n \in \mathcal{N}$. There exist planar HIST-critical graphs of order 3, 7, 11, 15, 17 and $3k+1$ for every even integer $k \geq 4$, while there are no such graphs of order $n \in \mathcal{N} \cup \{14\}$.



(a) G_k (b) $v = x$ (c) $v \in \{a_1, a_3, \ldots, a_{k-3}\}$

(d) $v = a_{k-1}$ (e) $v \in \{c_1, c_3, \ldots, c_{k-1}\}$ (f) $v \in \{c_2, c_4, \ldots, c_{k-2}\}$

Figure 6.1: Graph G_k and HISTs of vertex-deleted subgraphs of G_k.

Proof. We first prove the statements regarding the general (i.e. not necessarily planar) case. The exhaustive computations for HIST-critical graphs whose results were tabulated in Table 6.1 give the non-existence of HIST-critical graphs of order n for every $n \in \mathcal{N}$. By Theorem 6.2 and Proposition 6.3, for any non-negative integers k_1 and k_2 with $k_1 + k_2 \geq 2$ we obtain HIST-critical graphs of order $8k_1 + 12k_2 - (k_1 + k_2) = 7k_1 + 11k_2$. From the proof of Theorem 6.5 we obtain HIST-critical graphs of order $3k + 1$ for every even integer $k \geq 4$. It is elementary to verify that we thus obtain the theorem's first statement, using Table 6.1 and the subsequent remark on HIST-critical graphs of girth 8 and 9.

In a very similar way, the statement regarding planar HIST-critical graphs follows from Theorem 6.5, Table 6.2 and by verifying for Table 6.1 which graphs are also planar. This yields extra examples of order 15 and 17. □

It remains an open question whether there exists a HIST-critical graph of order n for $n \in \{26, 30, 34, 38, 45, 48, 52\}$.

6.4 On a conjecture of Malkevitch

Malkevitch conjectured in 1979 [97] that every planar 4-connected graph has a HIST. We computationally determined the following.

Proposition 6.7. *Every planar* 4*-connected graph up to and including order* 22 *has a HIST.*

Using `plantri` [21] we generated all planar 4-connected graphs up to order 22 and determined none of these were HIST-free using our algorithm in Section 6.2. The number of planar 4-connected graphs for each order can be found in Table E.1 in Appendix E.5. These counts extend the corresponding entry in the Online Encyclopedia of Integer Sequences which were previously only known up to 17 vertices (see: `https://oeis.org/A007027`).

It is natural to ask, if all of these graphs contain a HIST, how many HISTs such a graph should necessarily have. Denote by $p(n)$ the minimum number of HISTs in a planar 4-connected graph of order n. We summarise these counts in Table 6.3. For every entry there is always precisely one graph attaining the given number of HISTs.

For the even orders up to order 18 this minimum is attained by the antiprism (recall that antiprisms only exist for even orders) which motivates the following section, i.e. Section 6.4.1, where we establish the number of HISTs in an antiprism. A drawing of the graphs on odd orders attaining the minimum number of HISTs can be found in Figure E.6 in the Appendix.

n	6	7	8	9	10	11	12	13	14	15	16	17	18
$p(n)$	24	30	48	62	80	64	120	156	168	120	224	398	288

Table 6.3: Minimum number of HISTs in a planar 4-connected graph of order n.

6.4.1 Counting HISTs in antiprisms

An *antiprism* is a planar 4-connected even-order graph (V_k, E_k) with

$$\begin{aligned} V_k &= \{v_0, \ldots, v_k, w_0, \ldots, w_k\} \\ E_k &= \{v_0v_1, \ldots, v_kv_0, w_0w_1, \ldots, w_kw_0, v_0w_0, v_0w_1, v_1w_1, \\ &\quad v_1w_2, \ldots, v_kw_k, v_kw_0\}. \end{aligned}$$

For instance, the antiprism of order 6 is the octahedron.

Proposition 6.8. *The antiprism of order $2k$ with $k \geq 3$ has exactly $2k(2k-2)$ HISTs.*

Proof. We denote the vertex V_k and edge set E_k of the antiprism G of order $2k$ as above. Henceforth, we will assume that all indices are taken mod. k. We first show that a HIST in an antiprism cannot have a 4-vertex. Let T be a HIST of G containing a 4-vertex v. Then v must have one or two degree 3 neighbours. Since T is connected, a 3-vertex cannot have a degree 4 neighbour other than v in T and there is exactly one 4-vertex in T. Counting the degree sums, we see that the sum is a multiple of four if there is a 4-vertex, which implies that there is an even number of edges. However, since T is spanning its number of edges should be $2k-1$ which is a contradiction.

Similarly, using the degree sum, we see that a HIST T must have $k+1$ 1-vertices and $k-1$ 3-vertices. It is easy to see that the 3-vertices induce a subtree S of T, otherwise, T would not be connected. Moreover, S is an induced subgraph of G. Indeed, let u and v be two non-adjacent 3-vertices of T which share an edge in the antiprism. Let $u = v_i$ and $v = v_{i+1}$, then all incident edges of u and v except for uv are in T. Then since w_{i+1} has at least two incident edges, it is of degree 3 and either w_iw_{i+1} or $w_{i+1}w_{i+2}$ should be in T, but either of these lead to a cycle in T. It is straightforward to check that the other cases also lead to a cycle in T. It follows that S is an induced path of order $k-1$.

Let S be an induced path with endpoint w_i and either w_iv_i or w_iw_{i+1} lie in S. In order for v_{i-2} to be in T, it needs to be a 1-vertex, since such a path S can only include v_{i-2} if its order is k. Hence, either v_{i-3} or w_{i-2} lie in S. For the former case, we have $k-2$ possibilities. These paths are of the form $w_iw_{i+1}\cdots w_jv_jv_{j+1}\cdots v_{i-3}$, where $j = i, i+1, \ldots, i-3$. For the latter case, there is only one option, the path $w_iw_{i+1}\cdots w_{i-2}$.

Assume S to be of the form $w_iw_{i+1}\cdots w_jv_jv_{j+1}\cdots v_{i-3}$. In order for T to be a HIST, w_j must either have a neighbour w_{j+1} or v_{j-1} in T. Both of these

options completely fix T, hence there are precisely two HISTs for every such path S.

Let S be of the form $w_i w_{i+1} \cdots w_{i-2}$. In order for T to be a HIST, w_i must either have neighbours w_{i-1} and v_{i-1} in T or v_{i-1} and v_i. Both of these options completely fix T, hence we also have two HISTs in this case.

Letting the chosen endpoint be any of the $2k$ vertices of our antiprism, we get

$$2k(2(k-2)+2) = 2k(2k-2)$$

HISTs. Finally, note that the paths S with endpoint w_i and edges $w_i w_{i-1}$ or $w_i v_{i-1}$ are also paths of the form above but with a different endpoint. Hence, we can conclude that we counted all HISTs of the antiprism. □

6.4.2 4-regular graphs with or without HISTs

The fact that the triangle is HIST-free leads to the question whether other $2r$-regular HIST-free graphs exist; this problem was first formulated by Albertson, Berman, Hutchinson, and Thomassen [1]. They give an infinite family of 4-regular HIST-free graphs. Such a family of planar graphs had been given earlier by Joffe [84] but the relevant part of Joffe's thesis cannot be accessed, so hereunder we give an alternative (and short) proof of this result. We remark that the aforementioned question remains open for $r > 2$. In [1], they were particularly interested in the answer to the question whether 6-regular HIST-free graphs exist. They provide infinite families of $(2r+1)$-regular graphs for every natural number r.

Concerning HIST-critical graphs, we know that exactly one 2-regular such graph exists, and that no such graphs exist that are 3-regular. Whether k-regular HIST-critical graphs for $k > 3$ exist is unknown. Theorem 6.5 shows that there are infinitely many planar HIST-critical graphs with all but four vertices quartic. Unfortunately, we could not find a 4-regular HIST-critical graph, planar or not.

On the one hand, Malkevitch conjectured that every planar 4-connected graph has a HIST [97]. On the other hand, he remarked in [97, Remark 3], without giving a proof, that there exist planar 3-connected 4-regular graphs that are HIST-free. We now describe such graphs; in particular, we consider the line graph of cubic graphs.

Proposition 6.9. *There exist* 3*-connected* 4*-regular HIST-free planar graphs that are the line graph of cubic graphs.*

Lemma 6.10. *Let G be a cubic graph of order $4k+2$. Then there is a* 1*-to-*3k *correspondence between an induced tree T in G such that, for every edge in*

$E(G)\backslash E(T)$, precisely one of its ends lies on T, and the HISTs of the line graph $L(G)$. In fact, the set $E(T)$ coincides with the set of 3- or 4-vertices in a HIST of $L(G)$.

Proof. The line graph $L(G)$ has $6k+3$ vertices and is 4-regular where every edge is in precisely one triangle that surrounds a vertex in G. Let T' be a HIST of $L(G)$. By the same argument as in the HIST-freeness proof of Theorem 6.5, T' should have a 4-vertex and the other vertices are of degree 1 or 3. Let S be the edges of G corresponding to the 3- or 4-vertices in T'. Since T' does not have a cycle, S does not induce a cycle in G. Since every $e \in E(G)\backslash S$ corresponds to a 1-vertex in T', precisely one end of e is an end of an edge in S. Hence, S induces a tree in G with the desired property.

Next, let T be an induced tree of G such that for every edge in $E(G)\backslash E(T)$ precisely one of its ends lies on T. Let t be the order of T. Since any edge in $E(G)\backslash E(T)$ is incident to precisely one vertex in $V(G)\backslash V(T)$, we have $|E(G)| = (t-1) + 3(4k+2-t) = 6k+3$ and, hence, $t = 3k+1$. For any edge e in T, one can take a tree T' as a subgraph of $L(G)$ recursively: first T' is $K_{1,4}$ with the 4-vertex corresponding to e, and next add two pendant edges to a 1-vertex v in T' if v corresponds to an edge in T (note that the choice of two edges is unique since the adding of the other edge makes a triangle in T'). This recursive construction finally makes T' a HIST of $L(G)$. Note that T' does not depend on the order of 1-vertices v; see Figure 6.2 for example. Since $|E(T)| = 3k$, there are $3k$ choices of e and there is a 1-to-$3k$ correspondence between T and HISTs of $L(G)$. □

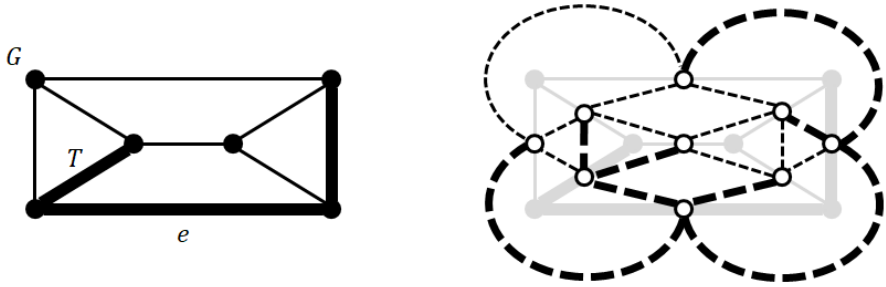


Figure 6.2: Left: induced tree T (bold edges) in a cubic graph G on which, for every edge in $E(G)\backslash E(T)$, precisely one of its ends lies. Right: HIST (dotted bold edges) in the line graph $L(G)$ with a 4-vertex corresponding to e.

Note that T gives a partition $V(T) \cup (V(G)\backslash V(T))$ of $V(G)$ such that $V(T)$ induces a tree of order $3k+1$ and $V(G)\backslash V(T)$ is an independent set of order $k+1$.

Let us illustrate the reasoning of Proposition 6.9 with an example. Let H_G be a truncated triangular prism; it is obtained from the cubic graph G depicted in the left of Figure 6.2 by replacing each vertex with a triangle. Then G is a cubic planar graph of order $18 = 4 \times 4 + 2$. Now H_G does not have an induced tree of order $4 \times 3 + 1 = 13$ since every set of 13 vertices should have three vertices of a triangle. By Lemma 6.10, we see that the line graph $L(G)$ does not have a HIST. Thus, $L(G)$ is a 3-connected 4-regular HIST-free planar graph. Similar examples can be easily constructed from any cubic planar graph of order $4\ell + 2$, letting G be its truncated graph.

Remark: In Lemma 6.10, if $|V(G)| = 4k$, then the following analogue holds which we state without proof. Here, a *unicyclic graph* is a graph with precisely one cycle. There is a 1-to-$6k$ correspondence between an induced unicyclic subgraph U of G such that, for every edge in $E(G)\backslash E(U)$, precisely one of its ends lies on U, and the HISTs of $L(G)$. In fact, the set $E(U)$ minus an edge coincides to the set of vertices of degree 3 in a HIST of $L(G)$. Note that U gives a partition $V(U) \cup (V(G)\backslash V(U))$ of $V(G)$ such that $V(U)$ induces a unicyclic graph of order $3k$ and $V(G)\backslash V(U)$ is an independent set of order k.

For the 4-connected case, the above method cannot be used to find a 4-regular HIST-free planar graph (i.e., a counterexample to Malkevitch's conjecture), due to the following argument and theorem.

Theorem 6.11 (Jaeger, [83])**.** *Let G be a connected cubic graph of order n and $s(G)$ the maximum number of vertices in a vertex-induced forest of G. Then*

$$s(G) \leq \left\lfloor \frac{3n-2}{4} \right\rfloor. \qquad (\dagger)$$

If the vertices of G can be covered by two vertex-disjoint vertex-induced trees of G, then equality holds in (†)*. As a corollary, if G^* is the dual of a hamiltonian maximal planar graph G, then equality holds in* (†)*.*

For a cubic graph G, if the line graph $L(G)$ is 4-connected, then G is cyclically 4-edge connected. Let G be a cyclically 4-edge connected cubic planar graph of order $n = 4k + 2$ (resp. $4k$). It is easily shown that the dual G^* of G is a 4-connected maximal planar graph; which is hamiltonian [133]. Then by Theorem 6.11, G has an induced forest F with $\lfloor \frac{3n-2}{4} \rfloor = 3k + 1$ (resp. $3k - 1$) vertices. When $n = 4k + 2$, we have $|E(F)| \leq 3k$ and $|E(G)\backslash E(F)| \leq 3(k + 1)$, both of which should attain the equality since $|E(G)| = 6k + 3$. This implies that F is a tree such that, for every edge in $E(G)\backslash E(F)$, precisely one of its ends lies on F. When $n = 4k$, we have $|E(F)| \leq 3k - 2$ and $|E(G)\backslash E(F)| \leq 3(k+1)$, which implies that one of the following cases occurs:

(a) F is a tree and $V(G)\backslash V(F)$ induces precisely one edge, say uv, or (b) F is a forest with precisely two components and $V(G)\backslash V(F)$ is an independent set. For (a), let F' be the graph induced by $V(F) \cup \{u\}$. For (b), let F' be a graph induced by $V(F) \cup \{w\}$, where w has neighbours in each of two components of F. (Such w exists since G is connected.) In both cases, F' is an induced unicyclic subgraph such that, for every edge in $E(G)\backslash E(F)$, precisely one of its ends lies on F. Thus, in both cases, it follows that for every cyclically 4-edge connected cubic planar graph, its (4-connected 4-regular planar) line graph has a HIST.

6.4.3 4-connected HIST-free graphs of small genus

We could not find a counterexample to Malkevitch's conjecture, so we tried to describe a 4-connected HIST-free graph of small genus. Here the (orientable) genus of a graph G is the smallest integer $g \geq 0$ such that G can be embedded on the orientable surface $\mathbb{S}_g$ of genus g. Let G be a 4-regular graph. The K_4-*inflation* of G is to replace each vertex v of G with K_4, and to join suitable two vertices of two K_4's so that the new edges are in 1-to-1 correspondence with the edges in G, see the top of Figure 6.3 of the image of a drawing.[2]

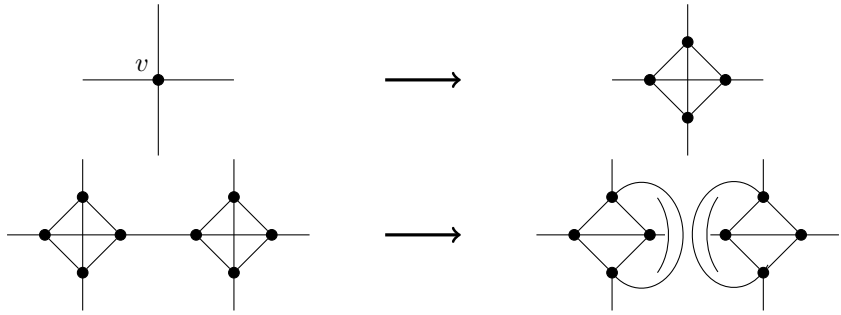


Figure 6.3: Top: replacing a vertex v with K_4. Bottom: adding a handle to decrease crossings by two.

Proposition 6.12. *Let G be a 4-regular graph without a hamiltonian path. Then the K_4-inflation H_G of G is HIST-free.*

Proof. Let v be a vertex of G and e_1, e_2, e_3, e_4 be the four edges incident to v. For a subtree T of H_G without 2-vertices, it is impossible that at least

[2]Given an abstract graph G, we here see a *drawing* of G as the mapping which assigns to each vertex a point of the plane and to each edge a simple continuous arc connecting the corresponding two points. When two such arcs intersect, we speak of a *crossing*. For an extensive discussion of drawings and crossings, see [107].

three edges of $\{e_1, e_2, e_3, e_4\}$ are in T if they are connected by the edges of K_4 corresponding to v. So it is not difficult to see that T corresponds to a subpath P of G. Since G has no hamiltonian path, P cannot span the vertices of G and T cannot span the vertices of H_G. □

Theorem 6.13. *There exists a* 4*-connected HIST-free graph of genus at most* 39*.*

Proof. Let G be the 3-connected 4-regular planar graph on 78 vertices, depicted in [134, Figure 11], which has no hamiltonian path. By Proposition 6.12, the K_4-inflation H_G of G is HIST-free. Since G is 2-connected 4-regular, it is not difficult to see that H_G is 4-connected.

We now give an upper bound for the genus of H_G. The image of a drawing of H_G has 78 edge crossings each of which corresponds to an inflated K_4. Adding a handle decreases crossings by two as depicted in the bottom of Figure 6.3. It is easy to find a perfect matching of G (of size 39), and adding handles along these 39 edges makes H_G an embedding on an orientable surface $\mathbb{S}_{39}$. □

Acknowledgements

We would like to thank Andreas Awouters for providing us with an independent implementation of the algorithm from Section 6.2 for counting the number of HISTs in a graph. Jan Goedgebeur and Jarne Renders are supported by Internal Funds of KU Leuven. Kenta Noguchi is partially supported by JSPS KAKENHI Grant Number 21K13831. Several of the computations for this work were carried out using the supercomputer infrastructure provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government.

Chapter 7

Network fault costs based on minimum leaf spanning trees

This chapter is a minimally edited version of the following submitted work.

J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. Network fault costs based on minimum leaf spanning trees. *arXiv preprint*, 2025. arXiv: 2502.10213

My contributions to this paper, in accordance with the contributor role taxonomy (CRediT)[1], consist of: Conceptualisation, Software, Validation, Formal Analysis, Investigation, Writing – Original Draft, Writing – Review and Editing, and Visualisation.

7.1 Introduction

We investigate the fault-tolerance of networks using spanning trees of the corresponding graphs. Optimisation problems concerning spanning trees occur in many applications, such as querying in computer database systems and connection routing. Consider a network modelled by a graph G. In various applications, in order to minimise costs, it is desirable to describe a spanning tree of G with as few leaves as possible. The number of leaves in such a tree is

[1] See https://credit.niso.org/.

called the minimum leaf number—the formal definition is given below. This number shall measure the quality of our solution: the smaller the number of leaves in a spanning tree, the better. In this article we want to investigate the situation when, due to a technical failure in the network, one of its nodes becomes unreachable. We model this by removing the corresponding vertex, which we call v, from G. In order to achieve a *fault-tolerant* network, we wish to guarantee that the minimum leaf number of $G - v$ is not larger than that of G, i.e. vertices in $G - v$ remain at least as well reachable as in G, and this holds true for an *arbitrary* node of the network as we do not know where the failure will occur. We shall here formalise this idea and give both theoretical as well as computational results. We will do so from two angles: on the one hand, we will investigate so-called leaf-guaranteed graphs, in which indeed vertices in $G - v$ remain at least as well reachable as in G, while on the other hand we shall introduce the notion of *fault cost*, a general tool to gauge how well a given arbitrary network performs with respect to an occurring fault (i.e., in our model, the loss of a node).

The vertex set and the edge set of a graph G is denoted by $V(G)$ and $E(G)$, respectively. The subgraph of G induced by $X \subseteq V(G)$ is denoted by $G[X]$ and let $G - X := G[V(G) \setminus X]$, $G - v := G - \{v\}$ for any $v \in V(G)$. For $v, w \in V(G)$ let $G + vw$ denote the graph obtained from G by adding the edge vw to $E(G)$. A graph is *connected* if there is a path between any two of its vertices. For an integer $k \geq 1$, if a connected graph G has a set of k vertices X for which $G - X$ has at least two connected components, we call X a *k-separator* of G. The graph G is *k-connected* if it has more than k vertices and does not have an m-separator for any $m < k$, and G is of *connectivity* k if it is k-connected and admits a k-separator or a set of k vertices whose removal leaves a single vertex. For vertices v and w in a graph, we write *v-path* for a path with end-vertex v, and *vw-path* for a v-path with end-vertex w. We denote the set of all spanning trees of G as $\mathcal{T}(G)$. A *leaf* is a vertex of degree 1, and, in a tree, a *branch* is a vertex of degree at least 3. A spanning tree with exactly k leaves is a *k-leaf spanning tree*. We call $L(G)$ the set of leaves of a graph G and put $\ell(G) := |L(G)|$. A vertex is *cubic* if it has degree 3, and a graph is *cubic* or 3*-regular* if all of its vertices are cubic. For a subgraph H of some given graph, if a vertex v in H has exactly k neighbours in H, we say that v has *H-degree* k. Throughout the paper, we assume graphs to be simple, undirected, and 2-connected, unless explicitly stated otherwise.

A graph on n vertices is *hamiltonian* if it contains a cycle of length n, i.e. a *hamiltonian cycle*, and it is *traceable* if it contains a path on n vertices, i.e. a *hamiltonian path*. We do not consider K_1 or K_2 to be hamiltonian. Following [140], the *minimum leaf number* $\mathrm{ml}(G)$ of a graph G is defined to be 1 if G is hamiltonian, ∞ if G is disconnected, and $\min_{T \in \mathcal{T}(G)} \ell(T)$ otherwise. It is

worth pointing out that the minimum leaf number of a graph is bounded above by its independence number, see [41, Proposition 8]. In a graph G, we will call a spanning tree or hamiltonian cycle S with $\mathrm{ml}(S) = \mathrm{ml}(G)$ an *ml-subgraph*. For further results on trees with a minimum number of leaves and equivalent problems, we refer to [41, 116, 12, 51], and for a pertinent US Patent of Demers and Downing, Oracle Corp., see [32].

A graph G is *k-leaf-guaranteed* if $k = \mathrm{ml}(G) \geq \mathrm{ml}(G - v)$ for all $v \in V(G)$. We denote the family of all k-leaf-guaranteed graphs with $\mathcal{L}_k$ and call a graph $G \in \bigcup_k \mathcal{L}_k$ *leaf-guaranteed*. It is easy to show (but we nonetheless give the proof in the next proposition, for completeness' sake) that $\{\mathrm{ml}(G-v)\}_{v\in V(G)} \subset \{k-1, k\}$ for any k-leaf-guaranteed graph G. We write $\mathcal{L}_k^\ell$ for the set of all graphs $G \in \mathcal{L}_k$ satisfying $\mathrm{ml}(G - v) = \ell$ for all $v \in V(G)$, where $\ell \in \{k-1, k\}$. Note that $\mathcal{L}_k \neq \mathcal{L}_k^k \cup \mathcal{L}_k^{k-1}$, since there exist graphs of which the vertex-deleted subgraphs have non-constant minimum leaf number.

Deciding whether a given graph has a certain minimum leaf number is an NP-complete problem, as it includes the hamiltonian cycle problem as a special case. Leaf-guaranteed graphs offer a common framework for a series of important graph families. $\mathcal{L}_k^k$ and $\mathcal{L}_k^{k-1}$ are exactly the *leaf-stable* and *leaf-critical* graphs, respectively, as investigated in [140]. These generalise hypohamiltonian and hypotraceable graphs and their applications include the solution to a problem of Gargano et al. [41] concerning the wave division multiplexing technology in optical communication. We recall that a graph is *hypohamiltonian* (*hypotraceable*) if the graph itself is non-hamiltonan (non-traceable) yet all of its vertex-deleted subgraphs are hamiltonian (traceable). Hypohamiltonicity—for an overview of theoretical results see the survey of Holton and Sheehan [78, Chapter 7]—has been applied in operations research in the context of the monotone symmetric traveling salesman problem [71, 72] as well as in coding theory [94]. Applications related to hypohamiltonicity also appear in the context of designing fault-tolerant networks, see [92, 138]. Moreover, various graph generation algorithms have been designed for hypohamiltonian graphs and related families [2, 63]. The family $\mathcal{L}_1^1$ is known as the 1-*hamiltonian* graphs, a classical notion in hamiltonicity theory [24]. In applications, these graphs are often called 1-*vertex fault-tolerant* [80, Chapter 12]. $\mathcal{L}_2 \cup \mathcal{L}_3^2$ are exactly the so-called *platypus graphs* [63, 145] which are connected to the Steiner-Deogun property [91] as described in [146].

In Section 7.2 we give structural properties of leaf-guaranteed graphs and then prove that for any network N on n nodes one can describe a leaf-guaranteed network with fewer than $16n$ nodes which contains N. In Section 7.3, the paper's main results are given. These revolve around the notion of a network's *fault cost*. We first give this new notion's formal definition and then give a

series of structural results. This is complemented by an algorithm, which we also implemented, to compute the fault cost. For instance, we determine for all non-negative integers $k \leq 8$ except 1 the smallest network with fault cost k. We also give a detailed treatment of the difficult fault cost 1 case, prove that there are infinitely many 3-regular networks with fault cost 3, and show that for any non-negative integer k there exists a graph with fault cost exactly k. The paper concludes with Section 7.4, in which we discuss open problems.

7.2 Leaf-guaranteed graphs

We begin by summarising some fundamental properties of leaf-guaranteed graphs.

Proposition 7.1. *Let $G \neq K_2$ be a k-leaf-guaranteed graph. Then the following hold.*

(i) *G is 2-connected, but not necessarily 3-connected, and the maximum degree of G is at least 3.*

(ii) $\mathrm{ml}(G - v) \in \{k - 1, k\}$ *for all $v \in V(G)$.*

(iii) *For any vertex v in G there exists an ml-subgraph T of G such that v is not a leaf of T. Moreover, for every k there exists a k-leaf-guaranteed graph G containing a vertex x which is not a leaf in any ml-subgraph of G.*

Proof. (i) Assume G has a 1-separator $\{x\}$. Then $G - x$ is disconnected, whence $\mathrm{ml}(G) = \infty$, a contradiction. Every leaf-stable graph and every leaf-critical graph is leaf-guaranteed. Leaf-stable and leaf-critical graphs of connectivity 2 were described in [106], so leaf-guaranteed graphs need not be 3-connected. Since a leaf-guaranteed graph is 2-connected but cycles are not leaf-guaranteed, its maximum degree must be at least 3.

(ii) Since $\mathrm{ml}(K_1) = 0$ and $\mathrm{ml}(K_2) = 2$, the assertion does not hold if $G = K_2$, although K_2 is leaf-guaranteed. We now prove the statement for every graph $G \neq K_2$ and assume this henceforth tacitly. We have that $\mathrm{ml}(G)$ is a positive integer or ∞ unless $G = K_1$. As G is leaf-guaranteed we have $\mathrm{ml}(G-v) \leq \mathrm{ml}(G)$ for any vertex v in G by definition, so the statement holds for all graphs G with $\mathrm{ml}(G) \leq 2$.

So let $\mathrm{ml}(G) = k \geq 3$. We continue with a proof by contradiction and suppose $\mathrm{ml}(G - v) \leq k - 2$ for some vertex v in G. If there exists a hamiltonian cycle

$\mathfrak{h}$ of $G - v$ we can easily modify $\mathfrak{h}$ to a hamiltonian path in G, so $\mathrm{ml}(G) \leq 2$, contradicting $\mathrm{ml}(G) = k \geq 3$. So $\mathrm{ml}(G - v) \geq 2$ certainly holds for all v in G.

Hence we may assume that there exists a spanning tree T of $G - v$ with at most $k - 2 \geq 2$ leaves. Let $w \in N(v)$. Then $\ell(T + vw) \leq k - 1$, so $\mathrm{ml}(G) \leq k - 1$, a contradiction. On the other hand, $\mathrm{ml}(G - v) \geq k + 1$ is impossible, since G is k-leaf-guaranteed, so by definition $\mathrm{ml}(G - v) \leq k$.

(iii) Both statements are obviously true for $k = 1$ (as no leaves are present), so assume henceforth $k \geq 2$. In a k-leaf-guaranteed graph G, consider a spanning tree T with k leaves, one of which shall be v. Let w be adjacent to v in T and $u \neq w$ a vertex adjacent to v in G (u exists as G is 2-connected by Proposition 2). Add the edge uv to T, resulting in the graph T'.

If u is a leaf in T, denote by s the vertex adjacent to u in T. Removing us from T' we obtain a new spanning tree T_0 of G. In T as well as T_0 the vertex u is a leaf, but v is not a leaf anymore in T_0, so T_0 certainly does not have more leaves than T. In T_0 the vertex s must be a leaf, as otherwise T_0 would be a spanning tree of G with fewer than k leaves, which is absurd. So T_0 is a spanning tree of G with $\ell(T) = \ell(T_0)$ and v is a leaf of T_0.

Consider now the situation that u is not a leaf in T. Remove from T' the unique edge lying on the unique cycle of T' and incident with u but not v. We obtain a new spanning tree T_1 of G in which v is not a leaf. Since $\ell(T) = k = \mathrm{ml}(G)$ the inequality $\ell(T_1) < \ell(T)$ is impossible, so $\ell(T_1) = \ell(T)$.

For the second statement, consider the Petersen graph P and two adjacent vertices v and w in P. It is well-known that both P and $P - w$ contain a hamiltonian path with an endpoint in v, a property we will refer to as $(\star)$. (The former follows from the fact that P is traceable and vertex-transitive, and the latter from the fact that P is hypohamiltonian.) Consider k pairwise disjoint copies $P^1, \ldots, P^k$ of $P - vw$, denoting the respective copies of v and w in P^i by v_i and w_i. We construct a graph G_k by identifying all v_i's, yielding one vertex x, and identifying all w_i's, yielding one vertex y. We shall see the P^i's as subgraphs of G_k.

Let T be a spanning tree of G_k. Since P is non-hamiltonian, T contains at least one leaf in each component of $G_k - x - y$. Therefore, $\mathrm{ml}(G_k) \geq k$. On the other hand, by $(\star)$, the graph G_k contains a spanning tree with exactly k leaves, whence, the minimum leaf number of G_k is precisely k. In order to prove that G_k is k-leaf-guaranteed, we need to show that $\mathrm{ml}(G - v) \leq k$ for every $v \in V(G_k)$. We need to differentiate between two cases. If $v \in \{x, y\}$, then we make use of $(\star)$ and obtain the desired conclusion. If $v \notin \{x, y\}$, then there exists an $i \in \{1, \ldots, k\}$ such that $v \in V(P^i)$. Since P is hypohamiltonian, there exists a hamiltonian x-path in $P^i - v$. Combining this with $(\star)$, we obtain

a spanning tree of $G_k - v$ with exactly k leaves. Thus, G_k is k-leaf-guaranteed. To conclude the proof, we will show that there exists no spanning tree T in G_k which has k leaves, and one of the leaves of T is x. However, as noted above, T contains at least one leaf in each component of $G_k - x - y$, of which there are k by construction. Therefore, T has at least $k+1$ leaves, a contradiction. □

It is natural to ask whether, given a network N, we can find a larger network N' containing a copy of N (or, in graph-theoretical terms: N is an induced subgraph of N') such that N' is leaf-guaranteed. It is known that this is possible if N' may be 40 times larger than N: in [147] it was shown that any graph G of order at most n, connected or not, occurs as induced subgraph of some hypohamiltonian and thus 2-leaf-guaranteed graph of order $40n$. Relaxing the conditions and only requiring containment in a leaf-guaranteed graph, we can describe significantly smaller solutions.

Theorem 7.2. *Let $G \neq K_1$ be a possibly disconnected graph of order n containing a longest path $\mathfrak{p}$ which has p vertices. Put $k := 2n - p + 1$. Then G occurs as an induced subgraph of some 1-leaf-guaranteed graph of order $k+1$ as well as of a k-leaf-guaranteed graph of order $8k$.*

Proof. Consider a k-cycle $C = v_1 \dots v_k$ disjoint from G. Put $\mathfrak{p} = w_1 \dots w_p$ and $V(G) \setminus V(\mathfrak{p}) = \{w_{p+1}, \dots, w_n\}$. Identify v_i with w_i for all $i \in \{1, \dots, p\}$ and v_{p+2j} with w_{p+j} for all $j \in \{1, \dots, n-p\}$. We have produced a graph H, where the purpose of the identification we just performed is to guarantee that G is indeed an induced subgraph of H.

For the first part of the statement, consider the graph H' of order $2n-p+2 = k+1$ obtained by taking H and an additional vertex v_0, and joining by an edge v_0 to v_i for all $i \in \{1, \dots, k\}$. That H' is 1-leaf-guaranteed is easy to see: Considering C now as a subgraph of H', we can replace the edge v_1v_2, which lies in C, by the path $v_1v_0v_2$ to obtain a hamiltonian cycle in H'. The cycle C itself is a hamiltonian cycle in $H' - v_0$. Consider v, an arbitrary vertex in $H' - v_0$, and let u and w be v's neighbours on C. Adding to the uw-path $C - v$ the path uv_0w yields a hamiltonian cycle in $H' - v$. Finally, it is clear that G is an induced subgraph of H', so the first part of the theorem is proven.

Let Ξ_8 be the 8-vertex graph obtained by connecting two vertices by three edges, subdividing once any two of the three edges, and blowing up the two cubic vertices to triangles (see the right-hand side of Figure 7.1). For the theorem's second statement, replace in H every edge of C by a copy of Ξ_8, see Figure 7.1. We now show that the resulting graph H'' is k-leaf-guaranteed. Let $S \in \mathcal{S}_{\mathrm{ml}}(H'')$. It is easy to check that Ξ_8 contains no hamiltonian vw-path,

where v and w are vertices as shown in Figure 7.1. Thus S must contain at least one leaf in each copy of Ξ_8 present in H'', whence $\ell(S) \geq k$.

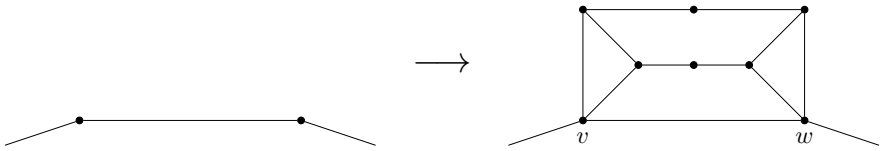


Figure 7.1: Replacing an edge with a copy of the graph Ξ_8.

It is straightforward to construct a tree with exactly k leaves in H'', so $\mathrm{ml}(H'') = k$. It remains to prove that $\mathrm{ml}(H'' - v) \leq k$ for every $v \in V(H'')$. By construction, we can restrict ourselves to an arbitrary but fixed copy of Ξ_8 in H''. It is easy to check that for every vertex u in Ξ_8, there exists a hamiltonian path in $\Xi_8 - u$ with at least one endpoint in $V(\Xi_8) \cap V(C)$; see Figure F.1 in Appendix F.1. As before, we can complete this path to a spanning tree of $H'' - u$ with exactly $2k$ leaves. We have proven that $\mathrm{ml}(H'' - v) \leq k$ for every vertex v in H'', whence, H'' is k-leaf-guaranteed. □

We end this section with a brief remark regarding a question of Grötschel. He asked in [68, Problem 4.56] whether *bipartite* hypotraceable graphs exist, i.e. whether any member of $\mathcal{L}_3^2$ is bipartite; note that hypohamiltonian graphs, i.e. members of $\mathcal{L}_2^1$, cannot be bipartite. As there has been little progress on this question, we relax it and ask for bipartite leaf-guaranteed graphs. In the upcoming Section 7.3.2 we describe an algorithm to determine the fault cost of a graph, a notion we shall define later. For a given graph G it computes ml-subgraphs of G and $G - v$ for all $v \in V(G)$. A straightforward adaptation allows to verify whether a graph is leaf-guaranteed. We generate 2-connected bipartite graphs using `geng` [100] and then apply our program [55] to check whether the input graphs are k-leaf-guaranteed for a certain k. Our computations yield that up to order 15, there is exactly one 2-connected bipartite leaf-guaranteed graph of order 12 (see Figure 7.2) and five more 2-connected bipartite leaf-guaranteed graphs on order 14. Restricting to girth at least 6 up to order 22 there is this aforementioned graph on order 12 as well as six more leaf-guaranteed graphs on order 16; 27 on order 18; 815 on order 20; and 11 775 on order 22. All of them are 2-leaf-guaranteed bipartite graphs and hence are 2-leaf-stable, i.e. in $\mathcal{L}_2^2$. We did not find any k-leaf-guaranteed examples with $k > 2$.

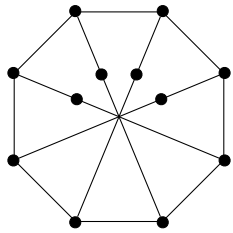


Figure 7.2: The smallest bipartite leaf-guaranteed graph.

7.3 Fault cost

7.3.1 Definition

We now introduce the notion of fault cost. From an application-oriented perspective it is important to point out that when a node drops from the network, the ml-subgraph used in the fault-free network may require changing equipment[2] in many nodes in order to obtain an ml-subgraph in the faulty network, which is undesirable. We formalise this by introducing, for a spanning tree or hamiltonian cycle S of G and a spanning tree or hamiltonian cycle S_v of $G - v$, the *transition cost from S to S_v* as

$$\tau(S, S_v) := |\{w \in V(G) \setminus \{v\} : \deg_S(w) \neq \deg_{S_v}(w)\}|.$$

Thus, this is the number of vertices in G which need to receive different equipment after the loss of a node; the lost node is ignored in this process. For a given graph G, denote by $\mathcal{S}_{\mathrm{ml}}(G)$ the set of all of its ml-subgraphs. In order to quantify the optimal solution in a worst-case scenario for the network itself, we introduce for a given graph G and an ml-subgraph S of G the quantity

$$\varphi_S(G) := \max_{v \in V(G)} \min_{S_v \in \mathcal{S}_{\mathrm{ml}}(G-v)} \tau(S, S_v).$$

Based on this, we shall consider the *fault cost* $\varphi(G)$ of the graph G representing the network:

$$\varphi(G) := \min_{S \in \mathcal{S}_{\mathrm{ml}}(G)} \varphi_S(G).$$

[2]We see vertices of different degrees as requiring different equipment in the network. As Salamon and Wiener write in [116], various problems related to ours have an objective function that depends on the vertex degrees of the spanning tree, see for instance [41, 95, 122]. This model is particularly useful when designing networks where device costs depend on the required routing functionality.

We give an example of how these definitions work in Appendix F.2. This framework is motivated by the fact that in a practical application one knows the network but must choose one of the ml-subgraphs without knowing which node might fail—so a subgraph ought to be chosen which minimises the maximum transition cost. We shall call such a subgraph, i.e. an ml-subgraph S in G satisfying $\varphi_S(G) = \varphi(G)$, *optimal*. Finally, when a vertex does fail, one again has the freedom to choose an ml-subgraph (in the faulty network) bearing the lowest transition costs[3].

7.3.2 Computational approach

An algorithm for computing the fault cost

Henceforth we focus on the description, both structural as well as computational, of graphs with small fault cost. For the latter approach we designed an algorithm to computationally determine the fault cost of a given graph. We use a backtracking approach and efficiently compute and store the distinct degree sequences of all ml-subgraphs and its vertex-deleted subgraphs. Once they have been determined, the degree sequences are compared and the fault cost determined. We use pruning rules, explained in more detail later in this section, in several places to speed up the computations. Our implementation of the algorithm is open source and can be found on GitHub [55]. An overview of the algorithm can be found in Algorithm 11.

In more detail our algorithm works as follows. Given an input graph G, we first check whether it is 1-hamiltonian using a program [56] designed by the authors for [53]. If the graph is 1-hamiltonian—clearly, there are infinitely many such graphs—, then we know by the following proposition that its fault cost is 0.

Proposition 7.3. *A graph has fault cost* 0 *if and only if it is* 1*-hamiltonian, i.e.* 1*-leaf-guaranteed.*

Proof. Recall that we work under the general assumption that our graphs are 2-connected, hence we do not treat K_2 (which is 1-leaf-guaranteed but not 1-hamiltonian). It is clear that a 1-hamiltonian graph G satisfies $\varphi(G) = 0$, so let us now assume that there exists a non-1-hamiltonian graph G with $\varphi(G) = 0$.

[3]In this model, it may happen that for a certain ml-subgraph only a few nodes are responsible for a high transition cost (and all other nodes yield far lower transition costs), but that for another ml-subgraph the transition costs are more evenly distributed and on average worse, but that the worst-case transition cost is in fact lower, so this latter tree will be chosen. This may not be desirable in certain applications, in particular when nodes fail with varying probabilities. However, in this article we are interested in conserving (or decreasing) the minimum leaf number whenever *any* vertex fails, irrespective of failure probability.

Algorithm 11 computeFaultCost(G)

```
if G is 1-hamiltonian then
    return 0
generateAndStoreMinLeafDegreeSequences(G) // Algorithm 12
for v ∈ V(G) do
    generateAndStoreMinLeafDegreeSequences(G − v)// Algorithm 12
for Sequence L of G do
    for v ∈ V(G) do
        for Sequence L_v of G − v do
            Compute transition cost
        Store minimum transition cost over L_v's
    Store maximum of minimum transition costs over v's
return φ(G)
```

The graph G cannot be hamiltonian, for otherwise there would exist a vertex x in G such that $G-x$ is non-hamiltonian. Thus, any ml-subgraph of $G-x$ would have at least two leaves, in which case the fault cost of G would be at least 2, a contradiction. Therefore G is non-hamiltonian and all of its vertex-deleted subgraphs must also be non-hamiltonian, similarly to the previous situation. Let now T be an optimal ml-subgraph of G. By the above arguments, T must be a tree. Consider a leaf b of T and let T_b be an arbitrary ml-subgraph of $G-b$. Again, by the above, T_b must be a tree. Then $\varphi(G)=0$ implies

$$\tau(T,T_b)=|\{v\in V(G)\setminus\{b\}:\deg_T(v)\neq\deg_{T_b}(v)\}|=0,$$

so

$$\sum_{v\in V(G)}\deg_T(v)=\deg_T(b)+\sum_{v\in V(G)\setminus\{b\}}\deg_{T_b}(v),$$

which, if we set $n:=|V(G)|$, is equivalent to $2(n-1)=1+2(n-2)$, a contradiction. □

We now continue with the description of our algorithm. If G is not 1-hamiltonian, we will determine all distinct degree sequences of all its ml-subgraphs (see Algorithm 12).

If the graph is hamiltonian (which we do not need to check again), its ml-subgraphs are the hamiltonian cycles of the graph. However, in that case there is only one distinct degree sequence, namely, the one in which every vertex has degree 2. If the graph is not hamiltonian, using methods from [56], we determine for every pair of non-adjacent vertices, whether or not there is a hamiltonian path between them. As the graph is not hamiltonian, there cannot

Algorithm 12 generateAndStoreMinLeafDegreeSequences(G)

```
Create 𝓛 // Data structure for storing the sequences
if G is hamiltonian then // Already known if we computed 1-hamiltonicity
    Store single sequence L where all degrees are 2
    return 𝓛
for each non-edge uv of G do
    if G has hamiltonian uv-path then
        Store sequence L, where u, v have degree 1 and x ≠ u, v has degree 2
if 𝓛 is not empty then // G is traceable
    return 𝓛
for each ml-subgraph of G do // Determined via backtracking and using pruning rules
    Compute degree sequence L
    if L ∉ 𝓛 then
        Store L in 𝓛
return 𝓛
```

be a hamiltonian path between adjacent vertices. For every non-adjacent pair we only need to determine one hamiltonian path as for every such hamiltonian path the degrees of every vertex except for the leaves are 2 and hence the degree sequences are the same. If we did not find any hamiltonian path in the graph, then we perform an exhaustive search for all ml-subgraphs as follows.

Using a backtracking algorithm, we generate all spanning trees of the graph. We start with one vertex and every iteration, we choose an edge incident to the current subtree which is not forbidden and first add it to the subtree (if it does not create a cycle), while later we forbid the edge. We then do the same for this newly acquired subtree or newly acquired set of forbidden edges. For the efficiency of the algorithm, we choose an edge for which the endpoint in the subtree v is most constrained. This means that, denoting the subtree by T and the subgraph of G induced by the forbidden edges by F, we want $d_G(v) - d_T(v) - d_F(v)$ to be minimal. Its other endpoint is the neighbour of v which is most constrained in the same way, such that the edge is not forbidden and not yet in the tree. We can backtrack whenever a vertex of G not yet in the tree is only incident with forbidden edges.

If the tree has size $|V(G)| - 1$, we have a spanning tree of our graph. If it has the same number of leaves as the minimum encountered so far, we store its degree sequence in a list. If it is smaller than the minimum encountered so far, we clear the list, store this degree sequence as its first entry and keep track of this new minimum. Since we know the minimum encountered so far, we can actually prune the search as soon as the subtree has more leaves than

this minimum. Due to the way we generate the spanning trees, the number of leaves can never decrease unless we backtrack.

In this way, we obtain all ml-subgraphs of G and we repeat this procedure for every vertex-deleted subgraph $G - v$. However, if v was a leaf of an ml-subgraph of G, we already know that $\mathrm{ml}(G)$ is an upper bound for the minimum leaf number of $G - v$ and keep track of this in order to prune earlier.

Now that we have every distinct degree sequence, we need to compare each degree sequence of the graph with each degree sequence of the vertex-deleted subgraphs. We also use some impactful techniques to speed up these computations. For every degree sequence of an ml-subgraph of G, we compare it to the distinct degree sequence of a vertex-deleted subgraph in order to determine the minimum transition cost. While checking the transition cost for two degree sequences is linear in the number of vertices, we also keep track of the leaves and degree 2 vertices of the graph, by means of bitsets. These low degree vertices can then be compared in constant time and we only need to iterate over the vertices of degree at least 3, i.e. the branches, in the ml-subgraphs. As the number of branches is relatively small (for example of the 153 620 333 545 graphs of order 12 which are 2-connected, only 144 of their ml-subgraphs contained four branches, while none contained more than four), the gain from this method outweighs the overhead of creating the extra bitsets and storing the degree 2 vertices in terms of efficiency. We also apply a small optimisation when iterating over the vertices. When the transition cost is already higher than one found earlier, we need not iterate over the remaining vertices. The minimum transition cost is then compared to those of other vertex-deleted subgraphs to determine the maximum. Finally, the minimum of these values is determined over all ml-subgraphs of G.

While it is relatively easy to prove the correctness of this algorithm, we also performed various tests in order to give evidence for the correctness of the implementation. A description of these tests can be found in Appendix F.3.

Computational results

We use our implementation of Algorithm 11 to obtain counts for the number of graphs attaining each fault cost for small graphs. We do this by generating all 2-connected graphs for a specific order using the program `geng`, which is part of the `nauty` library [100]. We then determine the fault cost for each of these generated graphs using our algorithm. Via `geng` it is easy and efficient to restrict the generation to graphs of at least a given girth (the *girth* of a graph G which is not a tree is the length of a shortest cycle in G) and go up to

higher orders in this way. The results of our computations[4] are summarised in Table 7.1.

$n\backslash\varphi$	0	1	2	3	4	5	6	7	8	9	10
3	0	0	1	0	0	0	0	0	0	0	0
4	1	0	2	0	0	0	0	0	0	0	0
5	3	0	7	0	0	0	0	0	0	0	0
6	13	0	43	0	0	0	0	0	0	0	0
7	116	0	341	0	11	0	0	0	0	0	0
8	2 009	0	5 016	6	92	0	0	0	0	0	0
9	72 529	0	119 730	130	1 677	0	0	0	0	0	0
10	4 784 268	0	4 926 191	3 930	29 141	0	12	0	0	0	0
11	554 267 470	0	345 785 155	133 243	783 029	47	137	10	0	0	0
12	111 383 671 391	0	42 204 241 063	6 480 547	25 933 650	2 112	4 98	184	0	0	0
13	258 028	0	8 890 460	62 453	980 464	425	1 114	114	3	0	0
14	11 388 066	0	233 751 383	1 326 967	16 543 622	7 023	17 890	1 236	13	2	0
15	62	0	13 7330	8 265	155 461	267	752	46	2	0	0
16	984	2	1 508 210	87 456	1 468 221	2 481	5 584	323	6	0	0
17	16 590	0	19 026 942	1 089 274	15 726 242	24 957	47 298	2 508	33	5	2
18	327 612	3	274 100 472	15 204 227	189 369 374	285 906	459 165	21 688	241	61	12

Table 7.1: Counts of how many 2-connected graphs attain each fault cost φ for each order n. The top part of the table gives counts for all 2-connected graphs, the middle part for 2-connected graphs of girth at least 4, and the bottom part for 2-connected graphs of girth at least 5. Fault costs for which the count is zero or which are not included in the table imply that no graphs of the given orders attain this fault cost, for example no graphs of order at most 12 have a fault cost higher than 7.

As can be seen from Table 7.1, graphs with fault cost 1 appear to be significantly rarer when compared to graphs with other low fault costs. Thus, after giving certain more general results, we will focus on structurally investigating fault cost 1 graphs in Section 7.3.4. In that section we present a 14-vertex graph with fault cost 1 (originally reported in the extended abstract [54]), see Figure 7.9. Combining this with the computational results given in Table 7.1, we obtain the following proposition, wherein φ_k shall be the order of the smallest graph with fault cost k.

Proposition 7.4. *We have* $\varphi_0 = 4, \varphi_1 \in \{13, 14\}, \varphi_2 = 3, \varphi_3 = 8, \varphi_4 = 7, \varphi_5 = 11, \varphi_6 = 10, \varphi_7 = 11$*, and* $\varphi_8 = 13$.

[4]These computations were performed on a cluster of Intel Xeon Platinum 8468 (Sapphire Rapids) CPUs. The computation for order 12 and no restriction on the girth took approximately 43 hours for the generation and 1 312 hours for computing the fault costs. For higher girths the fault cost computation is the bottleneck.

An example of a most symmetric graph attaining each φ_k can be found in the Appendix F.4. All of the most symmetric graphs attaining each φ as well as the smallest graphs[5] of girth 5 with fault cost 1 can also be searched on the House of Graphs [30] by using the keywords "fault cost".

Most of the families we deal with in the sequel are of connectivity 2. It is natural to ask whether for the same orders and fault-costs also 3-connected graphs appear. By filtering the graphs obtained from `geng` for 3-connectivity using the program `pickg` from the `nauty` [100] package, we can give a similar table for the fault costs of 3-connected graphs. See Table F.1 in Appendix F.5.

In order to obtain more examples at higher orders, we restrict our search to cubic graphs. We generated 2-connected cubic graphs using `snarkhunter` [19] and then determined the fault cost of each generated graph using our program. The results are summarised in Table 7.2.

As in the general case, it seems that also in the cubic case graphs with odd fault cost are a bit rarer than the graphs with even fault cost. In particular, the smallest cubic graphs with odd fault cost have order 22.[6]

As `snarkhunter` allows to restrict the generation to 3-connected graphs, we also determined their fault costs. The results are summarised in Table F.2 in Appendix F.6.

For planar graphs the most efficient way of generation depends on the connectivity of the graphs. We generated 2-connected planar graphs by generating all 2-connected graphs via `geng` and then filtered the planar ones using `planarg` which is also part of the `nauty` library [100]. We then used our program to determine the fault costs of these 2-connected planar graphs. The results are summarised in Table 7.3. For 3-connected planar graphs, generation can be done much more efficiently by using `plantri` [21]. We also applied our program to these graphs and the results are summarised in Table F.3 of Appendix F.7. Both in the cubic and in the planar case, the restriction to 3-connected graphs seems to restrict the fault costs, since, from the results in the appendix, it can be observed that for 3-connected cubic or planar graphs only fault costs 0 and 2 appear for small orders.

[5]The two graphs of order 16, girth 5 and fault cost 1 can be inspected on the House of Graphs [30] at `https://houseofgraphs.org/graphs/53049` and `https://houseofgraphs.org/graphs/53050`.

[6]The most symmetric cubic graph of order 22 with fault cost 3 can be inspected on the House of Graphs [30] at `https://houseofgraphs.org/graphs/53072`.

$n\backslash\varphi$	0	1	2	3	4
4	1	0	0	0	0
6	1	0	1	0	0
8	2	0	3	0	0
10	6	0	12	0	0
12	27	0	54	0	0
14	158	0	322	0	0
16	1 396	0	2 478	0	0
18	16 067	0	23 796	0	3
20	227 733	0	270 061	0	24
22	3 740 294	0	3 447 110	14	209
24	68 237 410	0	48 110 143	224	1 858
26	1 346 345 025	0	726 174 160	3 326	17 841
28	7 352 343 711	0	1 185 463 316	384	2 956
30	14 468 621 439	0	153 224 637	0	0

Table 7.2: Counts of fault costs for 2-connected cubic graphs. Fault costs for which the count is zero or which are not included in the table imply that no graphs of the given orders attain this fault cost. The top part of the table gives counts for all 2-connected cubic graphs, the middle part for 2-connected cubic graphs of girth at least 4, and the bottom part for 2-connected cubic graphs of girth at least 5.

$n\backslash\varphi$	0	1	2	3	4	5	6	7
3	0	0	1	0	0	0	0	0
4	1	0	2	0	0	0	0	0
5	2	0	7	0	0	0	0	0
6	7	0	37	0	0	0	0	0
7	34	0	249	0	11	0	0	0
8	246	0	2 549	6	92	0	0	0
9	2 526	0	32 396	119	1 455	0	0	0
10	30 842	0	489 870	2 682	22 402	0	12	0
11	416 108	0	8 138 729	59 733	414 982	38	137	10
12	5 955 716	0	143 994 243	1 385 013	8 223 342	1 327	3 734	184

Table 7.3: Counts of fault costs for 2-connected planar graphs. Fault costs for which the count is zero or which are not included in the table imply that no graphs of the given orders attain this fault cost.

7.3.3 Large fault cost

In Proposition 7.3 we characterised graphs with fault cost 0, establishing that these are exactly the 1-leaf-guaranteed graphs. Although we cannot give a characterisation, examples of graphs with fault cost 2 are easy to describe: hamiltonian, but not 1-leaf-guaranteed graphs are clearly of this type. However, they are not the only ones, e.g. $K_{2,3}$ also has fault cost 2, along with various other non-hamiltonian graphs. Deciding whether graphs with some given fault cost k exist is harder, even—or we might say especially—for $k = 1$. Therefore, in the sequel we shall focus on a structural description of fault cost 1 graphs. But before doing so, we give a general result relating the minimum leaf number of a graph to the minimum leaf numbers of the graph's vertex-deleted subgraphs, and prove that for every non-negative integer k there exists a graph with fault cost k.

Proposition 7.5. *Let G be a 2-connected graph. Then*

$$\mathrm{ml}(G) - 1 \leq \mathrm{ml}(G - v) \leq \mathrm{ml}(G) + \Delta(G)$$

for all $v \in V(G)$. Both bounds are best possible.

Proof. If G is hamiltonian we have $\mathrm{ml}(G) = 1$ and so $\mathrm{ml}(G - v) \in \{1, 2\}$ for all $v \in V(G)$. So the above inequalities trivially hold. Henceforth, we assume G to be non-hamiltonian, i.e. $\mathrm{ml}(G) \geq 2$.

Put $k := \mathrm{ml}(G)$. If there exists a $v \in V(G)$ such that $G - v$ is hamiltonian, i.e. $\mathrm{ml}(G - v) = 1$, then G is traceable and so $\mathrm{ml}(G) = 2$ since we are assuming that G is non-hamiltonian. Yet again the inequalities hold, so we may suppose that $G - v$ is non-hamiltonian for any $v \in V(G)$. Assume there exists a vertex $v \in V(G)$ such that $G - v$ contains a spanning tree T with at most $k - 2$ leaves. We add v and an edge incident to v to T and obtain a spanning tree of G with at most $k - 1$ leaves, a contradiction since $\mathrm{ml}(G) = k$. We have proven the lower bound and it is easy to produce examples exhibiting this bound, e.g. a hypotraceable graph.

Now we prove the upper bound. Let T' be an ml-subgraph of G. Note that T' is a tree since we are assuming G to be non-hamiltonian. Consider $v' \in V(G)$ with $\deg(v') = \Delta(G) > 1$. The disconnected graph $T' - v'$ is a forest consisting of pairwise disjoint trees $T_1, \ldots, T_{\Delta(G)}$. Since G is 2-connected, between any two trees T_i and T_j, $i \neq j$, there exists in $G - v'$ a path P between a vertex in T_i and a vertex in T_j such that $T_i \cup P \cup T_j$ is a tree. In an analogous manner we connect all trees $T_1, \ldots, T_{\Delta(G)}$ until a spanning tree T'' of $G - v'$ is obtained. As the tree T' has $\mathrm{ml}(G)$ leaves, the forest $T' - v'$ has at most $\mathrm{ml}(G) + \Delta(G)$ leaves. When constructing T'' in each step we add a path between two trees, so

in each step the total number of leaves cannot increase. Thus T'' also has at most $\mathrm{ml}(G) + \Delta(G)$ leaves. See Figure 7.3 for a graph G—indeed, an infinite family of graphs—showing that, in general, this bound cannot be improved: it is not difficult to check that $\mathrm{ml}(G) = \Delta(G)$ and $\mathrm{ml}(G - v) = 2\Delta(G)$ for v as defined in Figure 7.3. □

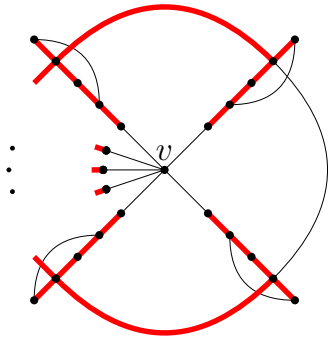


Figure 7.3: A 2-connected graph G with maximum degree and minimum leaf number $k := \deg(v) \geq 3$. In $G - v$, any spanning tree has at least $2k$ leaves. An ml-subgraph of $G - v$ with exactly $2k$ leaves is emphasised.

Theorem 7.6. *For any non-negative integer k there exists a graph with fault cost exactly k.*

Proof. We first describe a family of graphs with even fault cost $k \geq 4$. We handle the small and odd fault costs later. The family can be obtained by

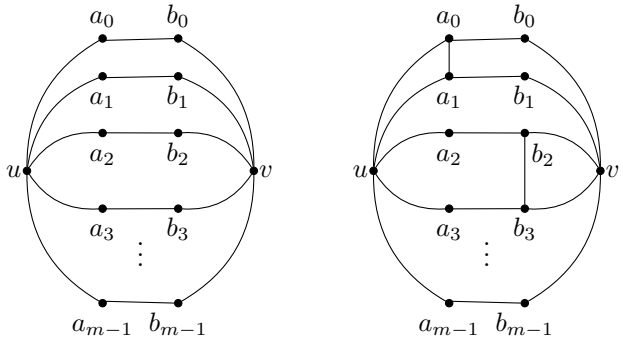


Figure 7.4: Left-hand side (a): The graph G_m, $m \geq 3$, with fault cost $2\lfloor m/2 \rfloor + 2$. Right-hand side (b): The graph H_m, $m \geq 5$ with fault cost $2\lfloor m/2 \rfloor - 1$.

taking the multigraph on two vertices and m edges between these two vertices, and subdividing each edge twice. See Figure 7.4(a). More formally, let $m \geq 3$ and let G_m be the graph with vertex set $V := \{u, v, a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1}\}$ and edge set $E := \{ua_i, vb_i, a_ib_i\}_{i=0}^{m-1}$. We now show that G_m has fault cost $m+2$ for even m and fault cost $m+1$ for odd m. In particular, the odd case gives us a graph with fault cost 4 when $m = 3$.

Let T be an ml-subgraph of G_m. We recall that for a vertex v in T, its *T-degree* is the degree v has in T. Since T is connected, there is a pair a_i, b_i with $i \in \{0, \ldots, m-1\}$ which both have T-degree 2. At most one such pair can exist as trees are acyclic; and since T is connected the remaining pairs a_i, b_i have one vertex of T-degree 1 and one vertex of T-degree 2. The number of leaves is minimised when u and v are not leaves in T. Hence, T has $m-1$ leaves. We define $\alpha := |L(T) \cap \{a_0, \ldots, a_{m-1}\}|$ and $\beta := |L(T) \cap \{b_0, \ldots, b_{m-1}\}|$. Without loss of generality, we assume that a_0 and b_0 are both of T-degree 2. We will now look at the ml-subgraphs of the vertex-deleted subgraph $G_m - x$ of G_m, where $x \in V(G_m)$.

Suppose that $x = u$. Then $G_m - x$ is a tree with m leaves and hence this tree T_x is the only ml-subgraph $G_m - x$. We have that $\tau(T, T_x) = 2 + 2\beta$ as a_0 and v change degrees (since $\alpha \geq 1$) and for every pair a_i, b_i where b_i is a leaf in T, the degrees change. Similarly, when $x = v$, $\tau(T, T_x) = 2 + 2\alpha$. Note that $2 + 2\alpha \geq 4$ and $2 + 2\beta \geq 4$ for any $m \geq 3$.

For any other vertex x of G_m, $\tau(T, T_x) \leq 3$ for an ml-subgraph T_x of $G - x$ which minimises this value. Indeed, let $x = a_0$, then an ml-subgraph has $m-1$ leaves. One can be obtained from T by changing the degrees of b_0, u and a_i, where a_i was a leaf of T. Similarly, when $x = b_0$, one can be obtained from T using only three changes. When x is a leaf of T, an ml-subgraph exists with only one change and when x is a vertex of degree 2, not a_0 or b_0, an ml-subgraph exists with three changes (u, v and the remaining neighbour of x in T).

Therefore, $\varphi_T(G_m) = \max\{2 + 2\alpha, 2 + 2\beta\}$, which is minimised over T when $\alpha = \lfloor (m-1)/2 \rfloor$ and $\beta = \lceil (m-1)/2 \rceil$ or vice versa. Hence, $\varphi(G_m) = m + 1$ when m is odd and $\varphi(G_m) = m + 2$ when m is even.

We now handle odd fault costs using the previous family, but with two added edges. See Figure 7.4(b). More formally, let $m \geq 5$ and $H_m := G_m + a_0a_1 + b_2b_3$. Then H_m has fault cost $m-2$ when m is odd and $m-1$ when m is even, as we now prove.

Any ml-subgraph T of H_m has exactly $m-3$ leaves and contains the path $vb_0a_0a_1b_1$ or $vb_1a_1a_0b_0$, where a_0 and a_1 have T-degree 2, as well as the path $ua_2b_2b_3a_3$ or $ua_3b_3b_2a_2$, where b_2 and b_3 have T-degree 2 and T has a pair of vertices a_i, b_i, with $i \in \{4, \ldots, m-1\}$, which are both of T-degree 2. Indeed,

in a spanning tree T' of H_m the set of vertices $\{a_i, b_i\}_{i=4}^{m-1}$ contains at least $m-5$ leaves of T', and at least $m-4$ leaves when there is no pair a_i, b_i, with $i \in \{4, \ldots, m-1\}$, such that both a_i and b_i have T'-degree 2. The set $\{a_i, b_i\}_{i=0}^{3}$ contains at least two leaves of T', but would introduce a cycle if $\{a_i, b_i\}_{i=4}^{m-1}$ contains a pair a_i, b_i such that both a_i and b_i have T'-degree 2, and one of a_0, a_1, b_2, b_3 would be of T'-degree 3. Therefore, an ml-subgraph T has the structure described above.

Due to symmetry, we can assume that T contains the paths $vb_0a_0a_1b_1$ and $ua_2b_2b_3a_3$ and that a_4 and b_4 are both of T-degree 2. We define α and β as before.

We now look at the ml-subgraphs T_x of the vertex-deleted subgraphs $H_m - x$ of H_m. Suppose $x = u$; again any ml-subgraph T_x will contain either $vb_0a_0a_1b_1$ or $vb_1a_1a_0b_0$. By our assumption on T, the option which minimises $\tau(T, T_x)$ always takes the former path. There are three options left for T_x as b_2 and b_3 can either both have T_x-degree 2; or b_2 can have T_x-degree 3 and b_3 can have T_x-degree 2; or b_2 can have T_x-degree 2 and b_3 can have T_x-degree 3. However, to minimise the transition cost, both b_2 and b_3 need to be of T_x-degree 2. This gives $\tau(T, T_x) = 3 + 2(\beta - 1)$, since the degree changes in v, a_2, a_4 and every pair a_i, b_i with $i \in \{5, \ldots, m-1\}$ in which b_i was a leaf in T. Due to symmetry, if $x = v$ then the ml-subgraph T_x which minimises the transition cost gives $\tau(T, T_x) = 3 + 2(\alpha - 1)$. Note that when $m \geq 6$, the maximum of these values is at least 4 and that when $m = 5$, both of them are equal to 3.

For any other vertex x of H_m we have $\tau(T, T_x) \leq 4$ for some ml-subgraph T_x of $H_m - x$. When $m = 5$, we even have $\tau(T, T_x) \leq 3$. Indeed, let $x = a_0$. Then an ml-subgraph of $H_m - a_0$ has $m-2$ leaves. One can be obtained from T by changing the degrees of u and b_0. The same holds for b_2. If $x = a_1$, an ml-subgraph has $m-2$ leaves and one can be obtained from T by changing the degrees of a_0 and v. The same holds for b_3. If $x = b_0$, an ml-subgraph has $m-3$ leaves and one can be obtained from T by changing the degrees of a_0 and b_1. The same holds for a_2. If x is a leaf of T, then an ml-subgraph of $H_m - x$ can be obtained from T using one degree change in the neighbour of that leaf. If $x = a_4$, then an ml-subgraph of $H_m - x$ has $m-3$ leaves if $m \geq 6$ and $m-2$ leaves if $m = 5$. The former can be solved by attaching a leaf which is not b_1 or a_3 to u or v. This can be done using at most four changes. The latter case can be dealt with in two changes adding the edge a_0u. The same holds when $x = b_4$. Finally, if x is any of the remaining vertices, i.e. a_i or b_i with $i \in \{5, \ldots, m-1\}$ with T-degree 2, then one can find an ml-subgraph of $H_m - x$ by only changing the degree of u and v.

Therefore, $\varphi_T(H_m) = \max\{3 + 2(\alpha - 1), 3 + 2(\beta - 1)\}$, which is minimised over T when $\alpha = \lfloor (m-3)/2 \rfloor$ and $\beta = \lceil (m-3)/2 \rceil$ or vice versa. Hence,

$\varphi(H_m) = m - 2$ when m is odd and $\varphi(H_m) = m - 1$ when m is even.

This proves the statement for graphs with fault cost $k \geq 3$. An example of a graph with fault cost 2 can be found in Appendix F.2. A search for fault cost 1 graphs is handled in the sequel. Graphs with vanishing fault cost exist due to Proposition 7.3. □

7.3.4 Small fault cost

As above experiments show, among graphs with small fault cost, graphs with fault cost 0 or 2 are ubiquitous and graphs with fault cost 1 or 3 are rarer. In the next two sections we therefore focus on these two small odd fault cost cases.

Fault cost 3

In this section we describe a construction for obtaining graphs of fault cost 3 and show there exist infinitely many cubic graphs with fault cost 3.

Let H be a not necessarily 2-connected graph. We say H is a *Type 1* graph if it contains pairwise distinct vertices v, w, x, y such that all of the following hold.

(i) There is no hamiltonian vw-path in H;

(ii) There is a vx-path P and a wy-path Q with $V(P), V(Q)$ partitioning $V(H)$;

(iii) H has a hamiltonian v-path and a hamiltonian w-path whose other endpoint lies in $\{x, y\}$;

(iv) $H - v$ and $H - w$ each contain a hamiltonian path with one endpoint in $\{v, w\}$ and the other endpoint in $\{x, y\}$;

(v) $H - u$ has a hamiltonian vw-path for any vertex $u \in V(H) \setminus \{v, w\}$;

(vi) H has a 3-leaf spanning tree with v, w leaves and the remaining leaf and branch in $\{x, y\}$.

Let H' be a graph containing distinct vertices v, w such that there is a hamiltonian vw-path, and for every $u \in V(H') \setminus \{v, w\}$ the graph $H' - u$ admits a hamiltonian vw-path or a 3-leaf spanning tree with v and w as leaves. We say H' is a *Type 2* graph.

Theorem 7.7. *For any integer $k \geq 3$, any Type 1 graph of order n_0 and any k choices of Type 2 graphs of order $n_1, \ldots, n_k$, respectively, there is a graph G of order $\sum_{i=0}^{k} n_i$ with $\varphi(G) = 3$. Moreover, if the Type 2 graphs have the property that for any vertex y and v, w as defined above, for any hamiltonian vy-path there is no hamiltonian wy-path, or vice versa, then this also holds for $k = 2$.*

Proof. Let G be the graph obtained by the disjoint union of a Type 1 graph H_0, where we denote v, w, as defined above, by v_0, w_0, and $k \geq 2$ Type 2 graphs $H_1, \ldots, H_k$, where we denote v, w in H_i, as defined above, by v_i, w_i, by adding the edges $w_i v_{i+1}$ for $i \in \{0, \ldots, k-1\}$ and $w_k v_0$. For the sake of convenience, we define $H_{k+1} := \emptyset$. We prove that G has fault cost 3.

The ml-subgraphs of G are its hamiltonian paths. Indeed, G is non-hamiltonian since H_0 has no hamiltonian $v_0 w_0$-path by (i), but G is traceable, since every H_i, $i > 0$, has a hamiltonian $v_i w_i$-path and H_0 has disjoint v_0- and w_0-paths spanning all vertices of H_0 by (ii).

Let us fix some ml-subgraph $\mathfrak{p}$ of G and denote its leaves by x and y. Due to (i) at least one of its leaves is a vertex in H_0. We assume for now that both x and y lie in H_0 and look at the ml-subgraphs of vertex-deleted subgraphs $G - u$ of G.

Suppose $u \in \{v_0, w_0\}$. Then $G - u$ is not hamiltonian as we remove a vertex from a 2-separator of G. However, $G - u$ is traceable since $H_0 - u$ has a hamiltonian v_0- or w_0-path by (iv). Therefore its ml-subgraphs are hamiltonian paths. Depending on the choice of path and the choice of x and y, this means that either zero, one or three vertices of H_0 change degree in an ml-subgraph S_u of $G - u$. Since S_u has a leaf in H_1 or H_k, we get $\tau(\mathfrak{p}, S_u) \in \{1, 2, 4\}$. Note that $\tau(T, S_u) = 1$ can only happen when u is x or y since we assume that $x, y \in V(H_0)$.

Consider $u \in V(H_0) \setminus \{v_0, w_0\}$. Since $H_0 - u$ has a hamiltonian $v_0 w_0$-path by (v), $G - u$ is hamiltonian so we have $\tau(\mathfrak{p}, S_u) \in \{1, 2\}$ in any ml-subgraph S_u of $G - u$.

Suppose $u \in \{v_1, w_k\}$. Then $G - u$ is non-hamiltonian as we have removed a vertex from a 2-separator of G. Since H_i, $i \in \{1, k\}$, has a hamiltonian $v_i w_i$-path because it is a Type 2 graph, we also obtain a hamiltonian path in H_i by removing u. As H_0 has a hamiltonian v_0- or w_0-path by (iii), we get that $G - u$ is traceable and that its ml-subgraphs are the hamiltonian paths. Depending on the choice of path and the choice of x and y, this means that either one or three vertices of H_0 change degree in an ml-subgraph S_u of $G - u$. As S_u has a leaf in H_1 or H_k, we have that $\tau(\mathfrak{p}, S_u) \in \{2, 4\}$.

Assume $u \in \{v_i, w_i\} \setminus \{v_1, w_k\}$ for an arbitrary but fixed $i \in \{1, \ldots, k\}$. Then a vertex of H_i and of H_{i-1} or H_{i+1} will have degree 1 in an ml-subgraph of $G - u$. Since H_0 has no hamiltonian v_0w_0-path by (i), $G - u$ is not traceable. However, since H_0 has a 3-leaf spanning tree in which v_0 and w_0 are leaves by (vi), the ml-subgraphs of $G - u$ are trees with exactly three leaves. Similarly, we can find 3-leaf spanning trees of $G - u$ of which the restriction to H_0 is a hamiltonian v_0- or w_0-path of H_0 by (iii). An ml-subgraph S_u of $G - u$ either has one, two, three or four vertices in H_0 which change degree with respect to $\mathfrak{p}$, depending on the choice of 3-leaf spanning tree of H_0 (or hamiltonian v_0- or w_0-path of H_0) and the choice of x and y. Therefore, $\tau(\mathfrak{p}, S_u) \in \{3, 4, 5, 6\}$. We note that any 3-leaf spanning tree of H_0 yielding an ml-subgraph of $G - u$ must have v_0 and w_0 as leaves.

Finally, suppose $u \in V(H_i) \setminus \{v_i, w_i\}$ for an arbitrary but fixed $i \in \{1, \ldots, k\}$. Then $G - u$ is still not hamiltonian. If $H_i - u$ has a hamiltonian v_iw_i-path, then there is an ml-subgraph S_u of $G - u$ for which $\tau(\mathfrak{p}, S_u) = 0$. If there is no such path, then H_i has a 3-leaf spanning tree with leaves v_i and w_i, as it is a Type 2 graph, and there exist ml-subgraphs S_u in $G - u$ for which $\tau(\mathfrak{p}, S_u) = 2$.

We have now shown that for any hamiltonian path $\mathfrak{p}$ of G with both leaves in H_0 that $\varphi_{\mathfrak{p}}(G) \geq 3$.

It also holds that for any hamiltonian $x'y'$-path $\mathfrak{p}'$ with x' in H_0 and y' in H_1 or H_k we have that $\varphi_{\mathfrak{p}'}(G) \geq 3$. Note that at least one leaf must be in H_0 by (i).

Indeed, taking $u \in \{v_i, w_i\} \setminus \{v_1, w_k\}$ for an arbitrary but fixed $i \in \{1, \ldots, k\}$, an ml-subgraph S_u of $G - u$ has leaves in H_i and H_{i-1} or H_{i+1} and a leaf and a branch in H_0. If $k \geq 3$, we can take u such that y cannot be one of these leaves and we get $\tau(\mathfrak{p}', S_u) \geq 4$. If $k = 2$, we have by the extra assumption on H_1 and H_2 that we can take u such that y is not a leaf in any ml-subgraph S_u of $G - u$. Hence, $\tau(\mathfrak{p}', S_u) \geq 3$.

If we now take $\mathfrak{p}$ to be a hamiltonian path in G with leaves x, y for which (i)–(vi) hold. Then by the above we have $\varphi_{\mathfrak{p}}(G) = 3$.

As, by definition, $\varphi(G) = \min_{S \in \mathcal{S}_{\mathrm{ml}}(G)} \varphi_S(G)$, this shows that $\varphi(G) = 3$.

□

Corollary 7.8. *There exist infinitely many cubic graphs with fault cost* 3*.*

Proof. Consider an integer $k \geq 2$ and let H_0 be the graph of Figure 7.5. It is tedious but straightforward to verify that it is a Type 1 graph for the indicated v, w, x, and y. Let H_i, for $i \in \{1, \ldots, k\}$, be copies of the graph in Figure 7.5(b). Again it is easy to verify that each H_i is a Type 2 graph. The result follows by

Theorem 7.7. It also holds for $k = 2$ as the extra condition of the theorem is satisfied for the graph in Figure 7.5(b). □

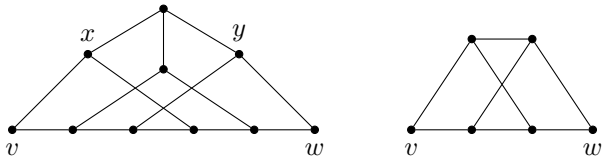


Figure 7.5: Subgraphs used in the proof of Corollary 7.8. Left-hand side (a): A Type 1 graph. Right-hand side (b): A Type 2 graph also satisfying the extra condition of Theorem 7.7.

Fault cost 1

When looking for graphs with fault cost 1, the first candidates to go over is the family of all graphs with minimum leaf number 2, that is: traceable, but not hamiltonian graphs. For traceable fault cost 1 graphs we will use the shorthand term *tfc1 graphs* in the sequel. Tfc1 graphs G must be 2-leaf-guaranteed: they cannot be hamiltonian, for otherwise the fault cost would be 0 or 2, showing that $\mathrm{ml}(G) = 2$. Now we have to prove that $\mathrm{ml}(G - x) \leq 2$ for all $x \in V(G)$. Assume this is not true for some x: then any spanning tree S_x of $G - x$ has at least 3 leaves and therefore there would also exist a vertex of S_x of degree at least 3, thus we would have at least 4 vertices of S_x not of degree 2, contradicting the tfc1 property of G. In fact, it is not difficult to see that in every tfc1 graph there exist at most two vertices whose deletion yields a hamiltonian graph:

Let G be tfc1. We already know that G is 2-leaf-guaranteed, so an optimal ml-subgraph P of G must be a hamiltonian path. Let the end-vertices of P be a_1 and a_2. If we delete a vertex v in G different from a_1 and a_2, we cannot obtain a hamiltonian graph, as then every ml-subgraph of $G - v$ would be a hamiltonian cycle and the degrees of a_1 and a_2 would be different in P and any ml-subgraph of $G - v$. This contradicts the fact that G has fault cost 1 or that P was an optimal ml-subgraph.

Claim 7.9. *Let G be a 2-leaf-stable graph. Then G has fault cost 1 if and only if there exist vertices $a_1, a_2 \in V(G)$, such that there exists a hamiltonian a_1a_2-path in G and for any vertex $x \in V(G) - a_1 - a_2$ there exists a hamiltonian a_1a_2-path in $G - x$ as well.*

Proof. Let P be a hamiltonian a_1a_2-path in G. We point out that P is an ml-subgraph of G because G is non-hamiltonian. In $G - a_i$ the hamiltonian

path $P - a_i$ is an ml-subgraph and $\tau(P, P - a_i) = 1$. For all $v \in V(G) \setminus \{a_1, a_2\}$, let M be any ml-subgraph of $G - v$. Then M is not a hamiltonian cycle as G is 2-leaf-stable. If M is chosen to be a hamiltonian a_1a_2-path, which we know exists, then $\tau(P, M) = 0$. So $\varphi_P(G) = 1$. Since G is not 1-hamiltonian, by Claim 1 we have $\varphi(G) = 1$.

In order to prove the other direction, let P be an optimal ml-subgraph of G. Since G is 2-leaf-stable, P is a hamiltonian path; let a_1, a_2 be the endvertices of P and let x be an arbitrary vertex in $V(G) - a_1 - a_2$. $G - x$ is non-hamiltonian, so there exists a minimum leaf spanning tree P' of $G - x$ (where P' is a hamiltonian path, since G is 2-leaf-stable), such that at most one of the vertices of $G - x$ has different degrees in P and P'. We show that the endvertices of P' are a_1 and a_2. Assume to the contrary that the endvertices are b, c, such that $\{a_1, a_2\} \neq \{b, c\}$, and w.l.o.g. also assume $a_1 \neq b, a_1 \neq c$. Then a_1 and (at least) one of the vertices b, c have different degrees in P and P', a contradiction. □

Graphs that are 2-leaf-stable and have fault cost 1 are called 2-*lsfc1* in the sequel.

Remark 7.10. *The vertices a_1 and a_2 do not have to be unique.*

An immediate corollary of the previous claim is that tfc1 graphs can have at most two vertices of degree 2 (by deleting a neighbour of a vertex $x \notin \{a_1, a_2\}$ of degree 2, we cannot have a hamiltonian a_1a_2-path). On the other hand, a_1 and a_2 might have degree 2 (and also any degree greater than 1), as we shall see later. This also means that tfc1 graphs need not be 3-connected. Now we are dealing with tfc1 graphs of connectivity 2, for which we need the following notions. Let X be a 2-separator of a graph G and let H be one of the components of $G - X$. Then $G[V(H) \cup X]$ is called a 2-*fragment* of G, and X is called the *attachment* of H. Let G_1 and G_2 be graphs, such that there exist two vertices x, y, such that $\{x, y\} = V(G_1) \cap V(G_2)$. Then $G_1 : G_2$ denotes the graph obtained by *gluing together* G_1 and G_2 at the vertices x, y, i.e. the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$. The next claim follows easily from Claim 7.9.

Claim 7.11. *Let G be a 2-lsfc1 graph, a_1, a_2 as described in Claim 7.9, and let $\{x, y\}$ be a separator of G. Then the following hold.*

(i) $\{a_1, a_2\} \cap \{x, y\} = \emptyset$.

(ii) *There are exactly two different 2-fragments G_1, G_2 of G with attachment $\{x, y\}$, such that $a_i \in V(G_i)$ for $i = 1, 2$.*

(iii) *There exists a hamiltonian a_ix-path in $G_i - y$ and a hamiltonian a_iy-path in $G_i - x$ for $i = 1, 2$.*

(iv) *For any $v \in V(G_i) \setminus \{a_i\}$ there exists a hamiltonian a_ix- or a_iy-path in at least one of the graphs $G_i - v$, $G_i - x - v$, $G_i - y - v$ for both $i = 1$ and $i = 2$.*

(v) *If $xy \notin E(G)$ then $G + xy$ is also a tfc1 graph.*

Note that statement (iii) of this claim is actually a special case of statement (iv), but it is worth mentioning it in its own right because of its corollaries.

We now focus on proving the existence of 2-lsfc1 graphs. In order to do so, let us observe that we may suppose that x and y are neighbours in a 2-fragment of a 2-lsfc1 graph, by point 5 of Claim 7.11. Assuming this, statement (iv) of Claim 7.11 becomes much easier to handle:

Claim 7.12. *Let $G, G_1, G_2, x, y, a_1, a_2$ be as described in Claim 7.11, such that $xy \in E(G)$. Then there exists a hamiltonian a_ix- or a_iy-path in G_i and also in $G_i - v$ for any $v \in V(G_i) \setminus \{a_i\}$ for $i = 1, 2$.*

It seems natural that a graph fulfilling the property described in Claim 7.12 is not necessarily a 2-fragment of some tfc1 graph. So we need further properties. Somewhat surprisingly, these properties are easy to describe and might not even be needed (at least not in both fragments).

Let H be a connected graph, $a, x, y \in V(H)$, and $xy \in E(H)$. Consider the following properties of the quadruple (H, a, x, y).

(P_0) For any $v \in V(H) - a$ there exists a hamiltonian ax- or ay-path in H and also in $H - v$.

(Q_1) There exists no hamiltonian xy-path in H.

(Q_2) For any $v \in V(H) - a$ there exists no hamiltonian xy-path in $H - v$.

The quadruple (H, a, x, y) is said to be a *weak fragment* if it fulfills (P_0), a *medium fragment* if it fulfills (P_0) and (Q_1), and finally a *strong fragment* if it fulfills (P_0), (Q_1), and (Q_2).

For convenience's sake, a graph H can also be called a weak/medium/strong fragment if there exist vertices $a, x, y \in V(H)$, such that (H, a, x, y) is a weak/medium/strong fragment. Note that medium fragments are also weak fragments and strong fragments are both medium and weak fragments as well. By gluing together such fragments we can obtain tfc1 graphs:

Theorem 7.13. *Let (G_1, a_1, x, y) and (G_2, a_2, x, y) be weak fragments. If both of them are also medium or one of them is also strong, then $G := G_1 : G_2$ is a tfc1 graph.*

Proof. G is easily seen to be traceable: there exists a hamiltonian a_1x- or a_1y-path P^1 in G_1, let us assume w.l.o.g. the former. There also exists a hamiltonian a_2x-path P^2 in $G_2 - y$. Now $P := P^1 \cup P^2$ is a hamiltonian a_1a_2-path of G.

The non-hamiltonicity of G follows immediately from the fact that (at least) one of G_1 and G_2 is medium and therefore there is no hamiltonian xy-path in (at least) one of G_1 and G_2.

Thus P is an ml-subgraph of G and it is enough to show that $\varphi_P(G) \leq 1$ (G is not hamiltonian, so $\varphi(G) \geq 1$ by Proposition 7.3). In order to do this we have to show that for each vertex $v \in V(G)$ there exists an ml-subgraph S_v of $G - v$ such that $\tau(P, S_v) \leq 1$.

Let us consider first the case $v = a_i$ for $i = 1, 2$. If $G - a_i$ is hamiltonian, then any ml-subgraph S_{a_i} of $G - a_i$ is a hamiltonian cycle, thus $\tau(P, S_{a_i}) = 1$. If $G - a_i$ is not hamiltonian, let $S_{a_i} := P - a_1$. Now $\tau(P, S_{a_i}) = 1$ is obvious again.

Now let $v = x$ (the case $v = y$ is the same). Since both G_1 and G_2 are weak fragments, there exist hamiltonian a_iy-paths P_i of $G_i - x$ for $i = 1, 2$. Now for the hamiltonian a_1a_2-path $P' := P_1 \cup P_2$ of $G - x$ we have $\tau(P, P') = 0$. Note that P' is an ml-subgraph of $G - x$, as $G - x$ is not hamiltonian (actually, not even 2-connected).

Finally, let $v \in V(G) \setminus \{a_1, a_2, x, y\}$. We show that $G - v$ is not hamiltonian. W.l.o.g. let us assume that $v \in G_1$. In order for $G - v$ to be hamiltonian we need a hamiltonian xy-path in both $G_1 - v$ and G_2. The existence of the former implies that G_1 is not a strong fragment, while the existence of the latter implies that G_2 is not a medium fragment, contradicting the choice of G_1 and G_2. Now we know that any hamiltonian path of $G - v$ is an ml-subgraph. Now by the weak fragment property of G_1 we have a hamiltonian a_1x- or a_1y-path P^1 of $G_1 - v$, w.l.o.g. let us assume that the endvertices of P^1 are a_1 and x. Since G_2 is also a weak fragment, there exists a hamiltonian a_2x-path P^2 of $G_2 - y$. Now for the hamiltonian a_1a_2 path $P' := P^1 \cup P^2$ of $G - v$ we have $\tau(P, P') = 0$, finishing the proof. □

Recall that in every tfc1 graph there exist at most two vertices whose deletion yields a hamiltonian graph. Note that the proof of Theorem 7.13 shows that in this construction only a_1 and a_2 might possess this property.

Weak fragments are easy to find, e.g. any complete graph of order at least 3 is a weak fragment; and by adding an edge to a non-complete weak fragment we also obtain a weak fragment. Examples are given in Figure 7.6. However, weak fragments are not enough to build tfc1 graphs using Theorem 7.13 as we need at least one medium fragment. This is somewhat harder to describe. Examples

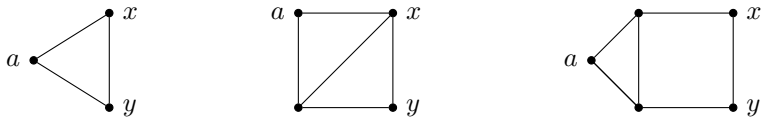


Figure 7.6: Examples of weak fragments.

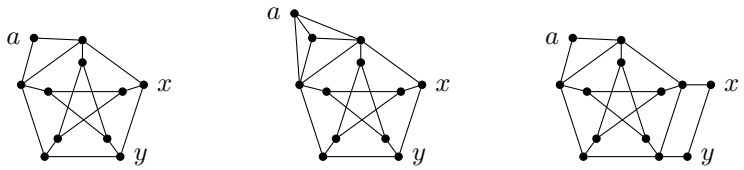


Figure 7.7: Examples of medium fragments.

based on Petersen's graph are shown in Figure 7.7 (we leave the verification that these are indeed medium fragments to the reader). Using two (not necessarily different) graphs of Figure 7.7 and Theorem 7.13, we obtain tfc1 graphs, see Figure 7.8.

Next we would like to find strong fragments, which is obviously even harder than finding medium ones. First we characterise 2-lsfc1 graphs with a separator $\{x, y\}$, such that x and y are neighbours. The next theorem shows that these can only be obtained in the way described in Theorem 7.13.

Theorem 7.14. *Let $G, x, y, a_1, a_2, G_1, G_2$ be as described in Claim 7.9 and Claim 7.11 and let $xy \in E(G)$. Then (G_1, a_1, x, y) and (G_2, a_2, x, y) are weak fragments and either both of them are also medium or one of them is also strong.*

Proof. By Claim 7.12, both (G_1, a_1, x, y) and (G_2, a_2, x, y) are weak fragments. Let us assume now that one of the fragments, say (G_2, a_2, x, y) is not medium, that is there exists a hamiltonian xy-path P^2 of G_2. In order to finish the proof we just have to show that (G_1, a_1, x, y) is a strong fragment, that is there is no hamiltonian path between x and y nor in G_1, neither in $G_1 - v$ for

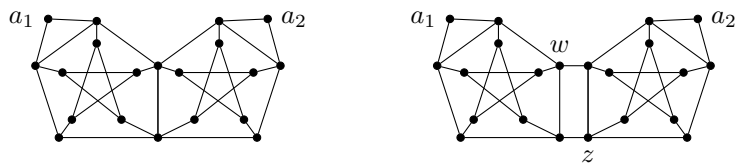


Figure 7.8: Examples of tfc1 graphs.

any $v \in V(G_1) - a_1$. Actually these are easy to see. The union of P^2 and a hamiltonian xy-path of G_1 would be a hamiltonian cycle of G, while the union of P^2 and a hamiltonian xy-path of $G_1 - v$ would be a hamiltonian cycle of $G - v$, both contradicting the 2-leaf-stability of G. □

Let us consider now (say) the first tfc1 graph of Figure 7.8 and its 2-separator X consisting of the neighbours of a_1. By Theorem 7.14 (which can be used, since the vertices in X are neighbours and $G - a_1$ and $G - a_2$ are easily seen to be non-hamiltonian), the 2-fragments with attachment X are weak fragments, and it is obvious that K_3 (one of the fragments) is not a medium fragment, therefore the other one (which is just the first graph of the figure with a_1 deleted) must be a strong fragment.

Corollary 7.15. *There exist infinitely many graphs with fault cost* 1*.*

Proof. We have seen that all complete graphs are medium fragments. Gluing these together with a strong fragment (whose existence we have just seen) we obtain infinitely many tfc1 graphs. □

If they exist, graphs of order 13 or 14 with fault cost 1 must, by Table 7.1, contain a triangle. Actually, we did find such graphs (of order 14, but none of order 13), as reported in the extended abstract [54] containing some of our initial results. In Figure 7.9 we reproduce one of these graphs. Using a computer it is easy to check the required properties, but—as it is stated, but not proved in [54]—there also exists a technique generalising Theorems 7.13 and 7.14, from which these immediately follow. These more general theorems are not included here, but might be subject of a follow up paper focusing mainly on the fault cost 1 case.

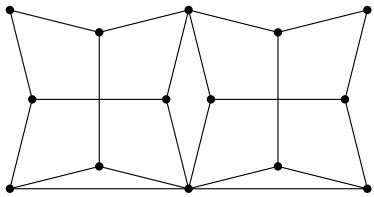


Figure 7.9: A 14-vertex graph with fault cost 1.

In this section we have assumed that the vertices of attachment are neighbours in a tfc1 2-fragment in order to make the construction of tfc1 graphs easier. However, there might be tfc1 2-fragments without this property, thus the

following questions arise naturally. Are there tfc1 graphs with a 2-separator consisting of non-neighbouring vertices? If so, do we have infinitely many? Are there tfc1 graphs, such that *all* 2-separators consist of non-neighbouring vertices? Again: if so, are there infinitely many?

The first question can be immediately answered in the affirmative: the separator $\{w, z\}$ of the second graph of Figure 7.8 possesses this property. This graph is obtained by gluing together the first and third medium fragments of Figure 7.7. Notice that we can create infinitely many medium fragments by (say) substituting a_1 and its two neighbours (that form a K_3) of the (say) first graph of Figure 7.8 with a different complete graph. This process works because of Theorems 7.13 and 7.14. By gluing together these medium fragments with the third medium fragment of Figure 7.7 we obtain infinitely many tfc1 graphs with a 2-separator of non-neighbouring vertices, answering the second question positively as well. Actually, we can even create tfc1 graphs with any given number of such 2-separators using the following straightforward lemma.

Lemma 7.16. *Let (H, a, x, y) be a weak fragment and let H' be the graph obtained from H by adding the vertices x' and y' and the edges $xx', x'y', y'y$. Then (H', a, x', y') is also a weak fragment, moreover if H is medium/strong then H' is also medium/strong.*

The third question can be answered affirmatively as well, since the graph of Figure 7.9 possesses the desired property. In order to obtain infinitely many such graphs however, we need more, like the aforementioned generalisations of Theorems 7.13 and 7.14.

7.4 Open problems

We end this paper with two natural problems on the structurally particularly challenging graphs with fault cost 1.

Problem 1. Up to now we have been discussing constructions based on 2-fragments, thus all of our tfc1 graphs are of connectivity 2. It is natural to ask whether 3-connected tfc1 graphs exist. If they do, is there a characterization for (say) the connectivity 3 case? To move even a bit further we might also ask whether k-leaf-guaranteed graphs with fault cost 1 exist for $k \geq 3$.

Problem 2. Are there cubic graphs with fault cost 1?

Acknowledgements

Several of the computations for this work were carried out using the supercomputer infrastructure provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government.

Chapter 8

Conclusions

In this section we reflect on the extent to which each chapter of the thesis has contributed to the central research objective of this thesis and summarise some of the directions for future work.

RQ: To what extent can computational methods contribute to extending the state-of-the-art research in structural graph theory on cycles?

We studied this question in six chapters that we subdivided into three parts based on how cycles appear in each of the topics. We looked at the interplay of computational and theoretical methods and tried to push this as much as possible in Chapters 2–7. Using the algorithms, we were able to discover certain properties about graphs, which we then proved theoretically. These properties in turn could often be used to speed up the algorithm, which allowed us to discover more properties, etc. We now go into more detail for each of these topics.

8.1 K_2-Hamiltonian Graphs

In Part I, Chapters 2 and 3, we studied K_2-(hypo)hamiltonian graphs and were the first to do so using computational methods. In Chapter 2, we developed a filter algorithm that determines for a given input graph whether it is K_2-hypohamiltonian. This algorithm proved to be quite useful for a first exploratory search into the graph class as we mapped out the presence of these graphs up to certain orders for general graphs, as well as up to higher orders in the

class of cubic graphs, snarks, planar graphs and cubic planar graphs. The main motivation for exploring many different graph classes was to push the algorithm as far as possible to find examples for which many vertex-deleted subgraphs are not hamiltonian, in an attempt to give more insight or maybe give an asymptotic answer to the variation of Grünbaum's conjecture [73] in the K_2-hamiltonian setting. For example, one can generate cubic planar graphs up to much higher orders, than one can for planar or cubic graphs. In particular, we determined all cubic planar K_2-hypohamiltonian graphs up to order 78, planar K_2-hypohamiltonian graphs up to order 48 and cubic K_2-hypohamiltonian graphs up to order 36. Moreover, these computations allowed us to obtain more theoretical results as the discovery of these graphs gave inspiration for coming up with an operation that preserves K_2-hypohamiltonicity, 3-regularity and non-3-edge-colourability, leading to infinite families and a complete characterisation of orders for which K_2-hypohamiltonian cubic graphs and snarks exist. We also discovered an operation preserving K_2-hypohamiltonicity and planarity leading to infinite families and an improvement of a result describing orders for which planar K_2-hypohamiltonian graphs exist. (The latter is only described in Chapter 3.) This already shows the merit of creating an algorithm that simply checks a property, but much of its success also comes from the many state-of-the-art generators with which we combined this method.

Despite these results, certain questions that could almost be answered with this approach lay just out of reach. For one, in the general case, all generated K_2-hypohamiltonian graphs were also hypohamiltonian. While it was known that this is not always the case, it is natural to ask for which orders this behaviour changes. This would improve the statement of Observation 2.12. Similarly, the characterisation of Theorem 2.11 remained incomplete. A slightly better algorithm, allowing us to generate all K_2-hypohamiltonian graphs up to order 17, would allow us to solve these questions. Since the current algorithm relies on the generation of general graphs and only filters the relevant examples and relatively few of these graphs are K_2-hypohamiltonian, a specialised generation algorithm would almost certainly perform better.

Due to the structural similarities with hypohamiltonian graphs and the work done there, we opted to take a similar approach, as was done for the generation of hypohamiltonian graphs [2, 59], as it seemed that we would be able to generate K_2-hypohamiltonian graphs up to order 17 using the specialised generation algorithm. This work was performed in Chapter 3. While the filter from Chapter 2 was implemented in a relatively straightforward manner, namely by applying a backtracking approach that tries to extend a path into a hamiltonian cycle and prunes as quickly and efficiently as possible, much more groundwork needed to be done in order to create an efficient generator. The idea also relies on a backtracking approach, but now we are building the graphs by adding

possible edges one by one and detecting when these graphs can no longer lead to a K_2-hypohamiltonian graph. In order to detect this efficiently, we needed to discover and prove certain properties of K_2-hypohamiltonian graphs that were relatively easy to compute, but which were ideally only present in very few non-K_2-hypohamiltonian graphs. This approach even allowed us to generate these graphs up to order 19 and we were able to solve the two previously mentioned questions. Moreover, the extra examples also allowed us to discover a new, but relatively easy to construct, infinite family of K_2-hypohamiltonian and hypohamiltonian graphs, which was previously unknown.

One may ask the question why it is even necessary to develop a filter if the specialised generator provides better results. Creating a specialised generator is of course more labour intensive than creating a filter. Moreover, one often needs to use the filter as a subroutine of the specialised generator as it still needs to check for the output graphs whether they are K_2-hypohamiltonian. Moreover, a filter together with a different state-of-the-art generator is also a good way of verifying the correctness of the newly created generator. Finally, one can say that creating better and better algorithms might yield diminishing returns. While in this case it was justified due to the open questions the faster/specialised generator would answer, if these questions would have been solved already by the generator-filter approach it might not have been worth the effort and therefore a first exploratory check using a generator-filter approach is desirable.

Our research has exposed several research lines for future work. One might try to answer the question whether K_2-hypohamiltonian bipartite graphs can exist. So far, there is no proof that they do not exist, but an extensive computational search did not yield any examples. In contrast, hypohamiltonian bipartite graphs cannot exist due to a simple argument. A second line of research is to consider the length of small cycles within K_2-hypohamiltonian graphs. No examples with girth at most 4 have been found, prompting the question of their existence. Similarly, examples with girth at least 7 are not known, but one might expect these to exist on much higher orders. Another line of research is to find operations on graphs preserving K_2-hypohamiltonicity, but creating graphs with few vertex-deleted hamiltonian subgraphs. So far only examples are known in which asymptotically a quarter of the vertex-deleted subgraphs are non-hamiltonian. We note that since we created a specialised generator for these graph classes, computational approaches for this work will need much faster methods or new theoretical insights improving the efficiency of the algorithm before significant improvements can be made.

8.2 2-Factors in Graphs

Part II contains Chapters 4 and 5. In Chapter 4, we developed a method for determining whether or not a given graph has Frank number 2. This question makes sense only in the case when the graphs are 3-edge-connected and due to the theoretical results of Chapter 4, the graph class of focus is the class of cyclically 4-edge-connected snarks as results in this graph class determine the behaviour of the Frank number of general graphs. While results on the theoretical time-complexity of this problem have been proven in [79], we were the first to implement an algorithm in order to study this problem using computational means.

The approach in this chapter is a good example of how theoretical results can improve the algorithm and how the algorithm can lead to new theoretical results. In first instance, we developed an exact algorithm, which due to the difficult nature of the problem, can only check relatively few graphs for this property in a feasible time. To speed up the algorithm we developed sufficient conditions for graphs having a Frank number of 2 in certain cases. These conditions can be checked a lot more efficiently and seemed to hold for more than 99% of the graphs we checked. Checking this condition first drastically sped up the algorithm. In turn, the results from the algorithm seemed to indicate that no graphs had a high Frank number. In our exhaustive search for the Frank number of cyclically 4-edge-connected snarks, *only* the Petersen graph had Frank number 3 out of 432 105 682 graphs, all others had Frank number 2. These facts prompted us to decrease the previously best known lower bound on the Frank number of 3-edge-connected graphs from 7 to 4. This was done in the cubic case by giving a construction of four nowhere-zero flows such that each edge attains value 1 in at least one of the flows, based on a decomposition of the graphs in terms of a $(\mathbb{Z}_2 \times \mathbb{Z}_3)$-flow. The existence of this flow is shown by Seymour's 6-flow theorem [119]. The cubic case then gives an upper bound also for general graphs.

Several lines of research are still open as possible future work. For one, no examples with Frank number 4 and no cyclically 4-edge-connected examples other than the Petersen graph with Frank number 3 are known. We conjecture they do not exist, but if they do, then a better algorithm, for example one which generates cyclically 4-edge-connected snarks up to orders higher than 36, might find one. As we mention in Chapter 4, the strong snarks and snarks with higher oddness are good candidates for counterexamples to our conjectures as here our sufficient conditions do not hold. We did not, however, implement a specialised generator for cyclically 4-edge-connected (cubic) graphs or snarks with Frank number at least 3. Partly, because we believe no such examples exist. Partly, because few properties of cyclically 4-edge-connected graphs with Frank number

at least 3 are known and they are necessary for ensuring such a specialised generator is efficient in practice and it would take a lot of time for very limited results (if we were to succeed in creating a significantly faster algorithm at all). Another open problem concerns the generalisation of the Frank number. The Frank number was developed in order to refine the connection between edge-connectivity of a graph and arc-connectivity of its orientations, but only when the graphs are 3-edge-connected. What about the case when k is odd and at least 5? One can come up with several ways to extend this notion. As an example, one might define a k-Frank number with $k \geq 1$, to be the minimum number of orientations in which any k-tuple of edges is deletable in at least one of them. Then the k-Frank number makes sense for $2k + 1$-edge-connected graphs and the 1-Frank number is precisely the original Frank number. However, the complexity of this problem increases drastically as k increases and hence computational methods will yield diminishing returns.

In Chapter 5, we developed a specialised generator for (non-hamiltonian) cycle permutation graphs and permutation snarks. In contrast to the previous topics, much more research as well as research using computational means has been done for these graphs. For example, in [89] non-hamiltonian cycle permutation graphs up to order 16 were generated and in [18] permutation snarks were generated using a generator-filter approach up to order 34. Using our algorithm, we were able to determine all non-hamiltonian cycle permutation graphs up to order 46, greatly improving on the results by Klee. Since the previous attempt to generate these graphs was made in 1972, one can wonder if our improved results are simply due to the increase in computing power. However, looking at the growth of non-hamiltonian cubic graphs[1], one sees that for small orders the number of graphs grows with a factor of approximately 8 every even order and this factor keeps growing. This very conservative estimation then says that in order to apply Klee's method to generate non-hamiltonian cycle permutation graphs up to order 46, computing power needs to have increased by a factor of 8^{15} or approximately 10^{13}. In this instance, our computations were performed on Intel's Ice Lake CPU's, with a max clock speed of 4.1 GHz. Klee did not mention the architecture he used, but his machine would certainly have been able to[2] perform more than 1 000 instructions per second and hence a very rough guess is that computation power increased by a factor of 4.1×10^6. This indicates that also the algorithmic efficiency has improved quite a lot and that faster computers alone are not enough to solve this problem, but also faster algorithms are necessary.

[1]The growth can for example be inspected on the Online Encyclopedia of Integer Sequences at `https://oeis.org/A164919`.

[2]IBM's first computer using transistors, the IBM 7070, could perform approximately 27 000 basic instructions per second. It was introduced 14 years before Klee's paper.

We were able to completely solve an open question by Klee asking for a characterisation of the orders for which non-hamiltonian cycle permutation graphs exist. We also generated permutation snarks up to order 46, which is a significant improvement of the results obtained by the generator-filter method of Brinkmann et al. [18], who were able to go up to order 34. One of the main motivations for studying these graphs is to gain more intuition in the class of permutation snarks, in particular the cyclically 5-edge-connected ones, due to their relationship with various conjectures. We succeeded in generating 736 extra cyclically 5-edge-connected permutation snarks, opening the door for further exploration of this class of which only 13 examples had previously been generated. In future work, one might classify these graphs according to their reason for non-3-edge-colourability and extend each of these classes into infinite families. A second motivation is to answer the question of existence of permutation snarks of order 6 mod 8. Our results show that a smallest one, if it exists, has order at least 54. This may indicate that such graphs do not exist, but future work studying the class of cyclically 5-edge-connected permutation snarks might bring more evidence for this.

Another line of future work is to test the newly obtained examples against certain conjectures in the hope of finding a counterexample. An example specific to cycle permutation graphs is by Goddyn [42]. He conjectures that the Petersen graph is the only cycle permutation graph without any *removable cycles* for any permutation 2-factor. In a cycle permutation graph G, a removable cycle C for a permutation 2-factor F is one such that $G - (E(C) \cap E(F))$ is 2-connected. This has been confirmed up to order 36 by Brinkmann et al. [18]. It is known that a counterexample, if it exists, is a permutation snark.

Finally, during our search, we did not find examples of permutation snarks with girth at least 6, despite the fact that cycle permutation graphs of arbitrary high girth can exist [109]. Showing they cannot exist or finding examples might yield more insight into this class.

8.3 Acyclic Graphs

Part III contains Chapters 6 and 7. In Chapter 6, we developed an algorithm for checking whether a given graph has a HIST or for counting the number of HISTs in a given graph. Slight alterations allow for filtering graphs which are HIST-critical, i.e. HIST-free, but every vertex-deleted subgraph has a HIST. The motivation for their study is to gain a better understanding of HIST-free graphs, which play an important role in the study of spanning trees. Using the results from our filter, we obtain a near-characterisation for the orders of HIST-critical

graphs, mirroring the study of hypohamiltonian graphs (hamiltonian cycles can be seen as antithetical to HISTs).

Our study is also the first to computationally verify a conjecture by Malkevitch stating that all 4-connected planar graphs have a HIST. While we did not find a counterexample, we showed it is true up to order 22, confirming Malkevitch's belief. This gives some evidence for the veracity of the conjecture. Our computational results indicate that in this class the antiprism graphs seem to have the fewest number of HISTs. We then proved a formula for their number of HISTs.

While we give a near-characterisation for which orders HIST-critical graphs exist, several orders between 26 and 52 remain unknown. A more specialised algorithm might make progress towards this question, but it seems a new theoretical approach is needed in order to handle the largest open cases, such as the discovery of operations creating HIST-critical graphs from smaller ones. However, as we observe in Section 6.3.2, a natural gluing procedure, which works well in the hypohamiltonian case, does not preserve HIST-freeness. Another question which remains open was asked by Albertson, Berman, Hutchinson and Thomassen [1] and asks for the existence even regular HIST-free graphs. While we give a proof for the existence of infinitely many examples in the case of 4-regular graphs, the question remains open for higher even regularities. Here, a specialised algorithm generating for example 6-regular HIST-free graphs, might yield more answers.

In Chapter 7, we developed an algorithm for determining the fault cost of a graph. As a consequence this also allows us to determine its minimum leaf number. Using the algorithm, we exhaustively determined the fault cost of all 2-connected graphs up to order 12. These results indicated that few small graphs have fault cost 1, prompting the theoretical question of whether more or infinitely many graphs of fault cost 1 exist. This is another example of how computational results can guide the direction of the research. An important exploratory question asks whether graphs with any fault cost can occur. We show this in Theorem 7.6. The infinite family used in this proof was discovered by studying examples that arose from the computational search. We also performed a similar exhaustive search in the cubic case. Here no graphs of fault cost 1 were found. Their existence is an interesting question for future research. Cubic graphs with fault cost 3 do not exist up to order 20, once the computer search found an example on order 22, however, we extended this into an infinite family of cubic graphs with fault cost 3.

One big open question is to study the behaviour of the fault cost for 3-connected graphs. It is for example unknown whether these graphs can have fault cost 1. A specialised generator might help in this direction.

8.4 Closing Remarks

This work applied computational methods to further the state-of-the-art research in structural graph theory in various topics concerning cycles. We show that this approach can successfully be applied to many topics in graph theory. However, one of its main limitations is that in order to obtain good results, the approach needs to be tailored to each specific problem and does not yield a general method that with little effort can be applied to all such problems.

Developing methods which are problem specific takes time, but we believe this investment is worth it as in our works we have significantly improved the state-of-the-art in each respective domain. This does not mean however that these algorithms are only useful for our results. In each of our papers, we try to be as transparent as possible with how the algorithm is implemented and make the code open-source. Moreover, we made interesting sets of graphs available online for example on The House of Graphs (see `https://houseofgraphs.org/`) to be used by other researchers. This might be most useful for generators. For example we have made abundant use of generators developed by other researchers in our works. However, also filter programs may be of use to other researchers. As an example, our program that checks K_2-hamiltonicity can also check whether or not a graph is hamiltonian and has been used in [61]. This shows that next to contributing directly to the state-of-the-art, we are also contributing indirectly by providing programs that might be used in future research.

Part IV

Appendix

Appendix A

Appendix to Chapter 2

A.1 Figures used in the proof of Lemma 2.2

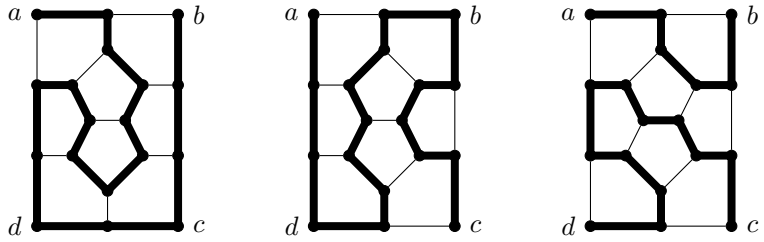


Figure A.1: Paths spanning J_{18}: a hamiltonian ab-path, a hamiltonian ac-path, and a disjoint ab-path and cd-path which together span J_{18}.

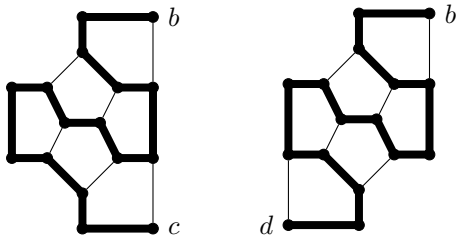


Figure A.2: A hamiltonian bc-path in $J_{18} - a - d$ and a hamiltonian bd-path in $J_{18} - a - c$.

A.2 Certificates for verifying the computer-aided proof of Lemma 2.3

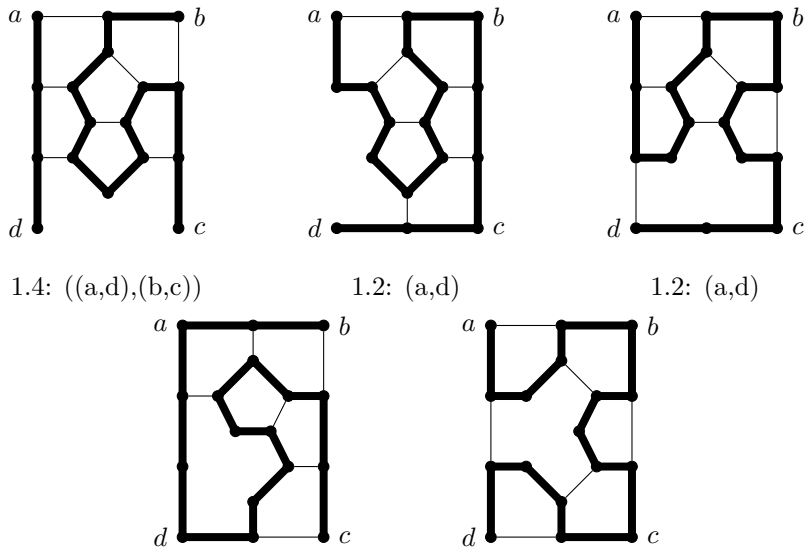
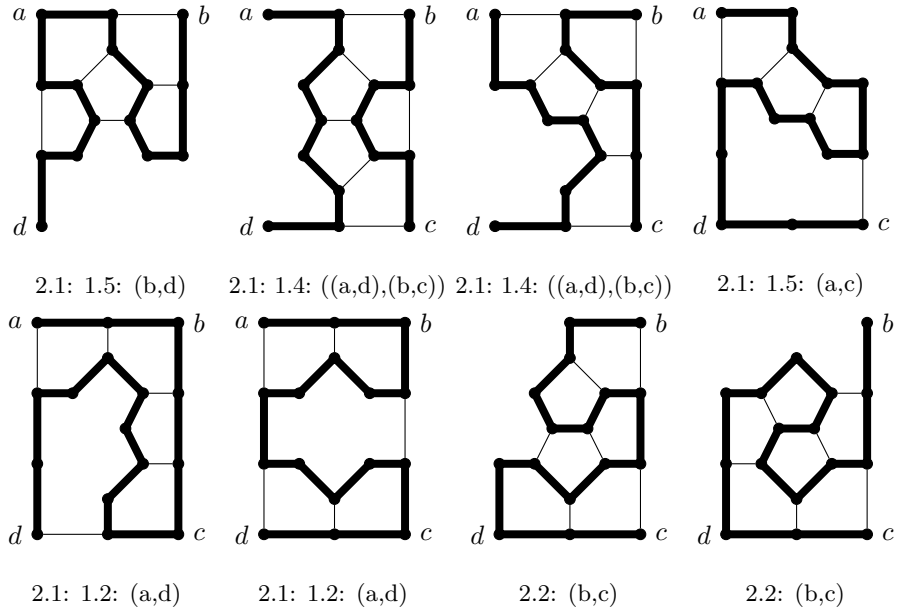


Figure A.3: Proof that J_{18} is a K_1-cell.



2.1: 1.5: (b,d) 2.1: 1.4: ((a,d),(b,c)) 2.1: 1.4: ((a,d),(b,c)) 2.1: 1.5: (a,c)

2.1: 1.2: (a,d) 2.1: 1.2: (a,d) 2.2: (b,c) 2.2: (b,c)

Figure A.4: Proof of properties 2.1 and 2.2-2.5.

A.3 Counts of K_2-hypohamiltonian graphs

Order	Girth	Total	Non-hamiltonian	K_2-hypoham.	hypo- and K_2-
5–9	≥ 3	273 183	2 500 918	0	0
10	≥ 3	11 716 571	2 411 453	1	1
11	≥ 3	1 006 700 565	123 544 541	0	0
12	≥ 3	164 059 830 476	11 537 642 646	0	0
13	≥ 3	50 335 907 869 219	2 013 389 528 672	1	1
14	≥ 4	445 781 050	297 831 030	0	0
15	≥ 4	13 743 625 184	8 160 280 174	1	1
16	≥ 4	566 756 900 370	290 053 195 532	4	4
17	≥ 5	179 593 823	167 592 079	0	0
18	≥ 5	2 098 423 758	1 923 066 160	3	2
19	≥ 5	28 215 583 324	25 294 705 218	28	28
20	≥ 5	434 936 005 284	379 901 735 283	2	2
21	≥ 5	7 662 164 738 118	6 490 985 473 090	31	28
22	≥ 6	38 259 063 075	37 983 633 685	0	0
23	≥ 6	342 877 495 737	340 026 856 312	0	0
24	≥ 6	3 340 738 751 508	3 307 741 496 902	0	0
25	≥ 7	91 618 528 203	91 585 941 929	0	0

Table A.1: Counts of all non-hamiltonian and K_2-hypohamiltonian graphs of a given order and girth. The last column lists the number of graphs that are both hypohamiltonian and K_2-hypohamiltonian.

A.4 All cubic planar K_2-hypohamiltonian graphs on 76 and 78 vertices

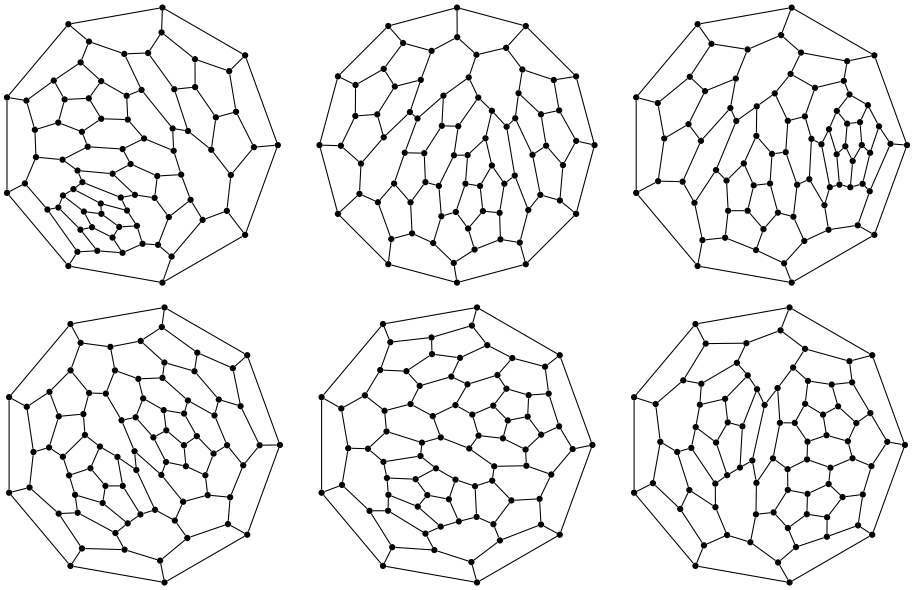


Figure A.5: The three cubic planar K_2-hypohamiltonian graphs on 76 vertices (top row) and the three such graphs on 78 vertices (bottom row), respectively.

A.5 Extendable 5-cycles

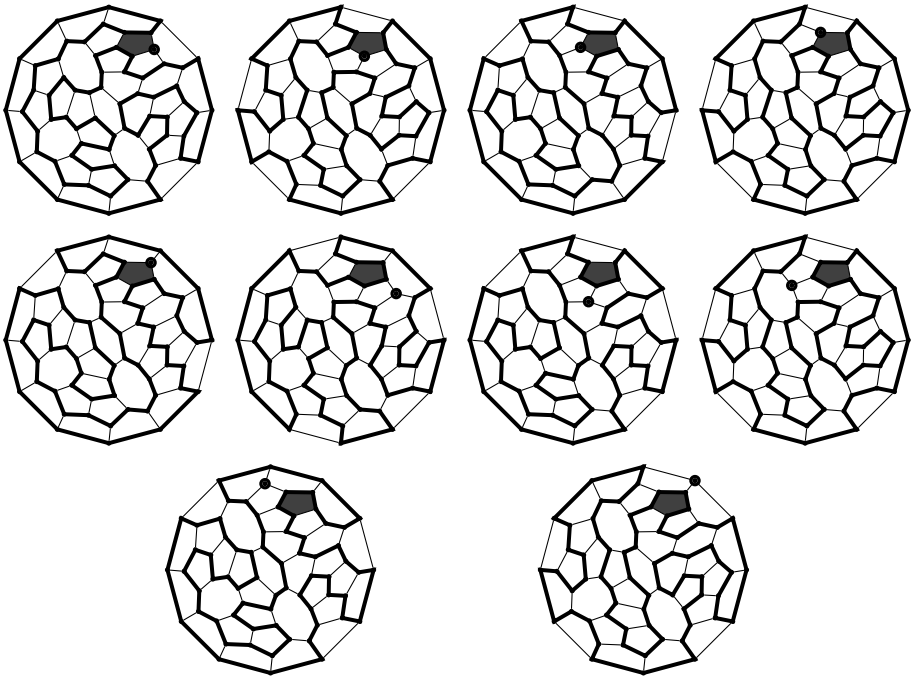


Figure A.6: A cubic planar K_2-hypohamiltonian graph and the proof that it contains an extendable 5-cycle.

Appendix B

Appendix to Chapter 3

B.1 Certificates for the proof of Theorem 3.22

For the proof of Theorem 3.22, we require four tuples satisfying the gluing property. Note that for readability, we removed all subscripts from the following figures and captions.

In Figure B.1, we see that $(G_{50}, a_{50}, a'_{50}, b_{50}, b'_{50})$ satisfies the gluing property, such that $G_{50} - a'_{50}$ has at least two hamiltonian cycles, one containing $b_{50}b'_{50}$ and one not containing it. Moreover, we see that in this embedding $a_{50}, a'_{50}, b_{50}, b'_{50}$ are co-facial. Also note that $G_{50} - v$ is hamiltonian for all $v \in \{a_{50}, a'_{50}, b_{50}, b'_{50}\}$ and that v is not among the vertices of the extendable 5-cycle and their neighbours.

In Figure B.2, we see that $(G_{52}, a_{52}, a'_{52}, b_{52}, b'_{52})$ satisfies the gluing property, such that $G_{52} - a'_{52}$ has at least two hamiltonian cycles, one containing $b_{52}b'_{52}$ and one not containing it. Moreover, we see that in this embedding $a_{52}, a'_{52}, b_{52}, b'_{52}$ are co-facial. Also note that $G_{52} - v$ is hamiltonian for $v \in \{a_{52}, a'_{52}, b_{52}, b'_{52}\}$ and that v is not among the vertices of the extendable 5-cycle and their neighbours.

In Figure B.3, we see that $(G_{52}, c_{52}, c'_{52}, d_{52}, d'_{52})$ satisfies the gluing property, such that $G_{52} - c'_{52}$ has at least two hamiltonian cycles, one containing $d_{52}d'_{52}$ and one not containing it. Moreover, we see that in this embedding $c_{52}, c'_{52}, d_{52}, d'_{52}$ are not co-facial. Also note that $G_{52} - v$ is hamiltonian for $v \in \{c_{52}, c'_{52}, d_{52}, d'_{52}\}$. However, v is not disjoint from the vertices of the extendable 5-cycle and their neighbours, but this is of no concern for the proof.

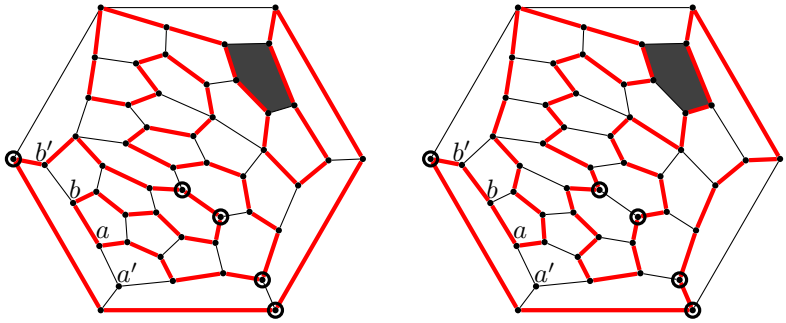


Figure B.1: The planar K_2-hypohamiltonian graph G_{50}. A face of which the boundary is an extendable 5-cycle is filled in. All vertices for which the vertex-deleted subgraph is non-hamiltonian are circled. On the left-hand side a hamiltonian cycle of $G_{50} - a'$ not containing bb' and on the right-hand side one which does contain bb' are marked by a thick line.

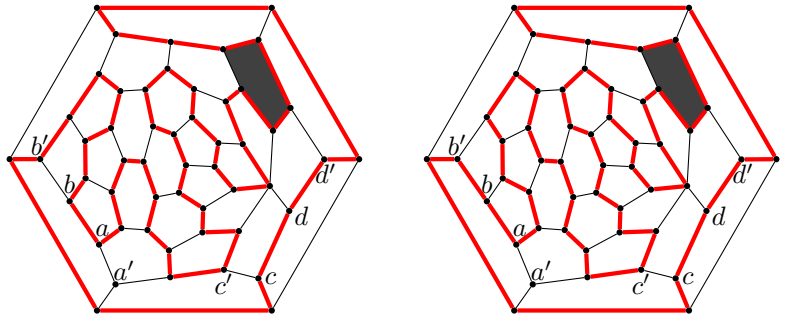


Figure B.2: The planar K_2-hypohamiltonian graph G_{52}. A face of which the boundary is an extendable 5-cycle is filled in. All of its vertex-deleted subgraphs are hamiltonian. On the left-hand side a hamiltonian cycle of $G_{52} - a'$ not containing bb' and on the right-hand side one which does contain bb' are marked by a thick line.

In Figure B.4, we see that $(G_{53}, a_{53}, a'_{53}, b_{53}, b'_{53})$ satisfies the gluing property, such that $G_{53} - a'_{53}$ has at least two hamiltonian cycles, one containing $b_{53}b'_{53}$ and one not containing it. Moreover, we see that in this embedding $a_{53}, a'_{53}, b_{53}, b'_{53}$ are not co-facial. Also note that $G_{53} - v$ is hamiltonian for $v \in \{a_{53}, a'_{53}, b_{53}, b'_{53}\}$ and that v is not among the vertices of the extendable 5-cycle and their neighbours.

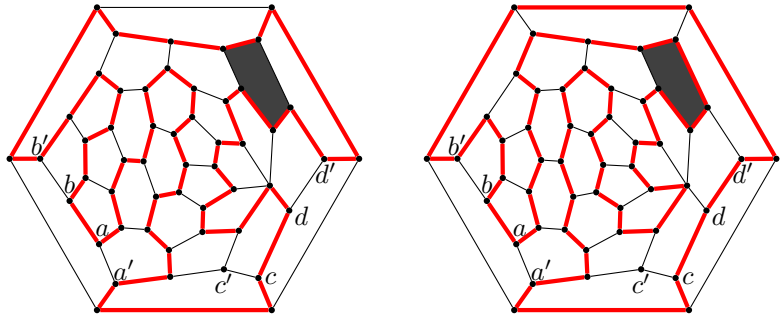


Figure B.3: The planar K_2-hypohamiltonian graph G_{52}. A face of which the boundary is an extendable 5-cycle is filled in. All vertices for which the vertex-deleted subgraph is non-hamiltonian are encircled. On the left-hand side a hamiltonian cycle of $G_{52} - c'$ not containing dd' and on the right-hand side one which does contain dd' are marked by a thick line.

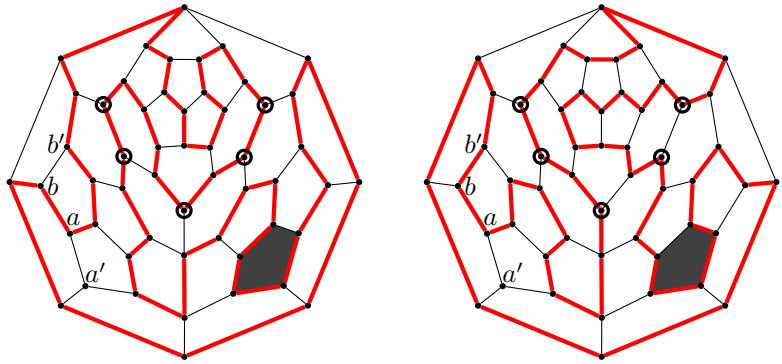


Figure B.4: The planar K_2-hypohamiltonian graph G_{53}. A face of which the boundary is an extendable 5-cycle is filled in. All vertices for which the vertex-deleted subgraph is non-hamiltonian are circled. On the left-hand side a hamiltonian cycle of $G_{53} - a'$ not containing bb' and on the right-hand side one which does contain bb' are marked by a thick line.

Appendix C

Appendix to Chapter 4

C.1 Algorithms

Algorithm 13 canAddArc(Partial Orientation (G, o'), Set D, Arc $u \to v$)

```
1: // Check if the addition of u → v will not violate rules of Lemma 4.15
2: if u has two outgoing arcs or v has two incoming arcs in (G, o') then
3:     return False
4: if uv ∈ D then
5:     for all edges e incident to u in G do
6:         if e ∈ D and e is an outgoing arc of u in (G, o') then
7:             return False
8:     for all edges e incident to v in G do
9:         if e ∈ D and e is an incoming arc of v in (G, o') then
10:            return False
11: else
12:    if in (G, o'), u has two incoming arcs or
       v has two outgoing arcs or
       u has an incoming arc whose corresponding edge is not in D or
       v has an outgoing arc whose corresponding edge is not in D then
13:        return False
14: return True
```

Algorithm 14 canOrientFixedEdges(Partial Orientation (G, o'), Set D, Arc $u \to v$)

```
1: // Recursively orient edges whose orientation is forced by Lemma 4.15
2: if u has two outgoing and no incoming arcs in (G, o') then
3:     Let ux be the edge of G which is unoriented in (G, o')
4:     if not canAddArcsRecursively((G, o'), D, x → u) then
5:         return False
6: if v has two incoming and no outgoing arcs in (G, o') then
7:     Let vx be the edge of G which is unoriented in (G, o')
8:     if not canAddArcsRecursively((G, o'), D, v → x) then
9:         return False
10: if uv ∈ D then
11:     if not orientDeletable((G, o'), D, u → v) then
12:         return False
13: else
14:     if not orientNonDeletable((G, o'), D, u → v)
        then
15:         return False
16: return True
```

Algorithm 15 orientDeletable(Partial Orientation (G, o'), Set D, Arc $u \to v$)

```
1: // Recursively orient edges whose orientation is forced by Lemma 4.15 in
   the case that uv ∈ D(G, o)
2: for all edges ux incident to u in G do
3:     if ux ∈ D then
4:         if not canAddArcsRecursively((G, o'), D, x → u) then
5:             return False
6: for all edges vx incident to v in G do
7:     if vx ∈ D then
8:         if not canAddArcsRecursively((G, o'), D, v → x) then
9:             return False
10: Let ux_1, uy_1 be the two edges incident with u in G such that x_1, y_1 ≠ v
11: if {ux_1, uy_1} ∩ D = ∅ then
12:     for z ∈ {x_1, y_1} do
13:         if not canAddArcsRecursively((G, o'), D, z → u) then
14:             return False
15: Let vx_2, vy_2 be the two edges incident with v in G such that x_2, y_2 ≠ u
16: if {vx_2, vy_2} ∩ D = ∅ then
17:     for z ∈ {x_2, y_2} do
18:         if not canAddArcsRecursively((G, o'), D, v → z) then
19:             return False
20: return True
```

Algorithm 16 orientNonDeletable(Partial Orientation (G, o'), Set D, Arc $u \to v$)

```
1: // Recursively orient edges which are forced by Lemma 4.15 in the case that uv ∉ D(G, o)
2: if u has precisely two incident arcs in (G, o') then
3:     Let ux be the edge of G which is unoriented in (G, o')
4:     if the arcs incident to u in (G, o') are one incoming and one outgoing then
5:         if not canAddArcsRecursively((G, o'), D, u → x) then
6:             return False
7: if v has precisely two incident arcs in (G, o') then
8:     Let vx be the edge of G which is unoriented in (G, o')
9:     if the arcs incident to v in (G, o') are one incoming and one outgoing then
10:        if not canAddArcsRecursively((G, o'), D, x → v) then
11:            return False
12: if there exists an edge ux of G such that x ≠ v and ux ∉ D then
13:    if not canAddArcsRecursively((G, o'), D, u → x) then
14:        return False
15: if there exists an edge vy of G such that y ≠ u and vy ∉ D then
16:    if not canAddArcsRecursively((G, o'), D, y → v) then
17:        return False
18: return True
```

C.2 Snarks for which Algorithm 3 is sufficient

Order	Total	Passed
10	1	0
18	2	1
20	6	6
22	31	29
24	155	152
26	1 297	1 283
28	12 517	12 472
30	139 854	139 547
32	1 764 950	1 763 302
34	25 286 953	25 273 455
36	404 899 916	404 793 575

Table C.1: Number of cyclically 4-edge-connected snarks for which Algorithm 3 is sufficient to decide that the graph has Frank number 2. In the second column the total number of cyclically 4-edge-connected snarks for the given order can be found. In the third column the number of such snarks in which the configuration of Theorem 4.11 or Theorem 4.13 is present is given.

Order	Algorithm 3	Remainder
28	4 s	9 s
30	42 s	304 s
32	585 s	3 h
34	3 h	106 h
36	54 h	4975 h

Table C.2: Runtimes of Algorithm 3 and 4 on the cyclically 4-edge-connected snarks of order 28 to 36. In the second column, the runtime of Algorithm 3 on these graphs can be found for the specific order. In the third column, the runtime of Algorithm 4 **only** on the graphs for which Algorithm 3 failed can be found.

Appendix D

Appendix to Chapter 5

D.1 Canonical construction path method

In this section we describe how to use the canonical construction path method, which is a generic method for the exhaustive isomorphism-free generation of graphs introduced by McKay in [99], in order to generate all pairwise non-isomorphic cycle permutation graphs. This method (if used correctly) guarantees that every graph is generated exactly once without having to store (and compare) all graphs in the memory during the generation process. Using similar ideas as in Section 5.2.2, the algorithm can also be adapted to only generate non-hamiltonian permutation graphs.

An *expansion* is an operation which constructs a larger graph from a given smaller one. The reverse operation is called a *reduction*. To use the canonical construction path method we will need to define a *canonical reduction* which is unique up to isomorphism and its inverse operation will then be the *canonical expansion*. Since our expansions will consist of adding a non-edge to the graph, we say two expansions applied to the same graph G are *equivalent* if there exists an automorphism of G mapping the respective non-edges to each other. This gives us, for each graph to which we apply expansions, classes of equivalent expansions.

When applying an expansion we need to determine whether or not we will accept the newly generated graph and keep adding edges to it or if we discard it and choose another edge to add in the smaller graph. In order to avoid isomorphic copies, we should accept every non-isomorphic intermediate graph exactly once. This can be done by following these two rules:

1. Only accept a graph if it was obtained by canonical expansion.
2. For every graph G to which expansions will be applied, only perform one expansion from each equivalence class of expansions.

In our case there is only one expansion operation, that is: adding an edge between pairs of specific vertices of degree 2.

Let G be a spanning subgraph of a cycle permutation graph which has a consecutive permutation 2-factor F with induced cycles C_1 and C_2. For $i \in \{1, 2\}$, if the degree 3 vertices on C_i induce a path P, let S_i be the set of vertices of degree 2 which are adjacent to P on C_i or let S_i be empty otherwise. We consider a pair of vertices *eligible* if one of the vertices lies in such a set S_i and the other is a vertex of degree 2 on C_{3-i}. We will apply the expansion to any eligible pair of vertices for any consecutive permutation 2-factor of our graph.

Note that the algorithm would also work if we add an edge between any pair of degree 2 vertices in which one lies on C_1 and the other on C_2. However, restricting the algorithm to eligible pairs drastically reduces the amount of times the recursive algorithm branches and hence makes the implementation a lot more efficient.

The high level pseudocode for the recursive method of the algorithm can be found in Algorithm 17.

Algorithm 17 Expand_CCPM(G)

```
if G is cubic then
    Output G
Determine all eligible pairs of vertices.
Determine the orbits of all such pairs.
for all eligible pairs (u, v) do
    // Cf. Rule 2 of the canonical construction path method.
    if (u, v) is the representative of its orbit then
        if adding edge uv to G is a canonical expansion then // Cf. Rule 1.
            Expand_CCPM(G + uv)
```

We now explain the algorithm in more detail. Suppose we want to generate the cycle permutation graphs of order n. We start with G_0 consisting of two cycles $C_1 = v_0v_1 \dots v_{n/2-1}$ and $C_2 = w_0w_1 \dots w_{n/2-1}$ and the edge v_0w_0. Note that $C_1 \cup C_2$ forms a consecutive permutation 2-factor. Let G be a (sub)cubic supergraph of G_0 of order n which contains a consecutive permutation 2-factor.

If G is cubic then it is a cycle permutation graph. It should be output and the recursion backtracks. If not, we determine all consecutive permutation 2-factors

in order to find the eligible pairs of vertices. In practice, we keep track of all consecutive permutation 2-factors dynamically for efficiency reasons. More specifically: after adding an edge, we remove those which have become non-consecutive or are no longer permutation 2-factors and search for consecutive permutation 2-factors containing the newly added edge. While constructing the induced cycles of such a 2-factor they are stored in a bitset, so that we can use bit operations to determine efficiently whether or not they have chords. This search for new consecutive permutation 2-factors containing the most recently added edge is one of the bottlenecks of the algorithm.

Given all consecutive permutation 2-factors, it is straightforward to determine all eligible pairs of vertices. We then need to determine the orbits of these pairs in order to only add an edge between one of the vertex pairs of each orbit, since all pairs in the same orbit would lead to isomorphic graphs. To obtain these orbits we use the program `nauty` [100] for determining all generators of the automorphism group of G. The orbits are then determined using a union-find algorithm. We choose the representative of each orbit to be the first vertex pair of that orbit we encountered while searching for eligible vertex pairs.

Once we have determined that an eligible vertex pair (u, v) is the representative of its orbit, we need to determine whether its addition to G will be a canonical expansion. In practice, we do this by looking at the graph $G + uv$ and analysing its *reducible* edges. These are the edges e for which $G + uv - e$ contains a consecutive permutation 2-factor. These are however easily obtained once one knows all consecutive permutation 2-factors of $G + uv$. Other edges are not reducible and can be ignored since removing them does not give us a graph which needs to be generated.

In order to determine whether or not we should accept $G + uv$ we need to define a canonical reduction which is unique up to isomorphism. The edge whose removal yields the canonical reduction will be the *canonical edge*. To this end, we assign a 10-tuple $(x_0, \ldots, x_9)$ to each reducible edge and let the canonical edge be the one with the lexicographically maximal value for this 10-tuple.

For a reducible edge $e = ab$, the values $x_0, \ldots, x_7$ are invariants of increasing discriminating power and cost, determined empirically. They are defined as follows:

- x_0 is the negative of the number of vertices at distance at most 2 from a or b.
- x_1 (x_2) is the negative of the number of 4-cycles (5-cycles) containing the edge ab.
- x_3 is the number of vertices at distance at most 3 from a or b.

- x_4 is the negative of the number of 6-cycles containing the edge ab.
- x_5 (x_6) is (the negative of) the number of vertices of degree 2 at distance 1 (at most 2) of a and of b.
- x_7 is the number of vertices at distance at most 4 from a or b.

We note that while the discriminating power is negligible for the later invariants, they are necessary for the efficiency of variants of this program, such as the generation of non-hamiltonian cycle permutation graphs or cycle permutation graphs with girth restrictions. See Section 5.3. Moreover, their presence here does not noticeably increase the running time of the algorithm.

While the above values are invariant under isomorphism, their values could be the same for non-isomorphic graphs. Therefore we define $\{x_8, x_9\}$ to be the lexicographically largest label of an edge which is in the same edge orbit as e in the canonical labelling of the graph. We again use `nauty` [100] for determining this canonical labelling of the graph. (This gives us the generators of the automorphism group for free, hence if we accept this expansion, we take care not to call `nauty` a second time.)

Note that in theory we could define the canonical edge using only x_8 and x_9, however, as calling `nauty` is computationally expensive, it is much more efficient to compute the other invariants first. Once such an invariant is able to show that our added edge uv is not canonical, for example if there is a reducible edge with fewer vertices at distance at most 2, then we can reject the expansion immediately and do not need to compute the remaining invariants (including the expensive call to `nauty`). Similarly, the invariants can be sufficient to determine if our added edge is the only reducible edge with maximum value for the tuple and hence it is canonical. The expansion can then immediately be accepted without computing the remaining invariants. As an example, on order 24, invariants $x_0, \ldots, x_7$ are sufficient to discriminate the canonical edge in 94.64% of the cases. If we were to only consider x_8 and x_9 for this order, this would slow down the total computation by a factor of 5.

One of the bottlenecks of our algorithm is the search for new consecutive permutation 2-factors containing the most recently added edge. This needs to happen after adding an eligible non-edge e in order to determine all reducible edges and find out whether the addition of e was a canonical expansion. We try to avoid this as much as possible. Suppose we are considering intermediate graph G and eligible non-edge e. One of the ways we avoid this search is by first only removing the 2-factors of G which have become non-consecutive or which are no longer permutation 2-factors in $G + e$ and determining a subset of the reducible edges based on this subset of consecutive permutation 2-factors of

$G + e$. One can then already compute the values $x_0, \ldots, x_9$ for this subset of reducible edges and one might be able to deduce whether e is non-canonical. If this is the case, we can prune here.

If e is not yet determined to be non-canonical then either it is canonical, or there is another consecutive permutation 2-factor yielding new reducible edges, one of which is a canonical edge. Before starting the search for these 2-factors, we first check the condition given by Proposition 5.4.

If we find that this condition holds, then we have already computed all reducible edges in the previous step and determined whether or not e is canonical. Otherwise, we will need to perform the algorithm for finding a consecutive permutation 2-factor containing e. We can do this in the same way as was done in Algorithm 7.

Theorem D.1. *For a given order n, Algorithm 17 will output all pairwise non-isomorphic cycle permutation graphs of order n exactly once.*

Proof. Let G be a cycle permutation graph of order n. We show that it will be output by the algorithm. Let G_m be a (sub)cubic subgraph of G containing a consecutive permutation 2-factor F with $n+1 < m \leq 3n/2$ edges such that if it (or a graph isomorphic to it) is accepted by the algorithm, then G (or a graph isomorphic to G) will be output. It is easy to see that if $G_{3n/2}$ is isomorphic to G and gets accepted that (a graph isomorphic to) G will be output by the algorithm. Since G_m contains F, it also contains reducible edges and one of them must be the canonical edge, say e. Hence, if a graph isomorphic to $G_m - e$ was accepted by the algorithm, adding the image of e under the isomorphism will be the canonical expansion, and a graph isomorphic to G_m will be accepted. Finally, we complete the argument by noting that $G_{n+2} - e$, where e is the canonical edge of G_{n+2}, is isomorphic to the starting graph of our algorithm, which shows that G_{n+2} will be accepted.

Conversely, we show that if G is output by the algorithm it is the only graph of its isomorphism class which is output. Suppose that G' is also output and is isomorphic to G. Let

$$G_{n/2+1}, G_{n/2+2}, \ldots, G_{3n/2} = G$$

be the sequence of graphs with consecutive permutation 2-factors where G_m has m edges and $G_{m-1} = G_m - e$ where e is the canonical edge of G_m which were accepted by the algorithm to eventually output G. Define

$$G'_{n/2+1}, G'_{n/2+2}, \ldots, G'_{3n/2} = G'$$

similarly. Then either, for all $m \in \{n/2+1, \ldots, 3n/2\}$ we have that G_m is isomorphic to G'_m. However, since $G_{n/2+1} = G'_{n/2+1}$ and there must be an

m' for which $G_{m'} \neq G'_{m'}$ as labelled graphs, we have performed an expansion using an eligible vertex pair which was not the representative of its orbit, which gives a contradiction. Hence, we have that there must be an m' such that G_m is isomorphic to G'_m for all $m' < m \leq 3n/2$, but with $G_{m'}$ and $G'_{m'}$ not isomorphic. Let $G_{m'+1} - e = G_{m'}$ and $G'_{m'+1} - e' = G'_{m'}$, then both e and e' must be canonical edges in their respective graphs and since $G_{m'+1}$ is isomorphic to $G'_{m'+1}$ there must be an isomorphism mapping these edges onto each other. However, this isomorphism then gives rise to an isomorphism between $G_{m'}$ and $G'_{m'}$, which is a contradiction. Hence, we conclude that the algorithm will only output one graph of each isomorphism class. □

D.2 Correctness testing

Next to proving the correctness of the algorithms, it is also important to verify the correctness of their implementations. To this end, we created a second implementation for generating all pairwise non-isomorphic (non-hamiltonian) cycle permutation graphs and permutation snarks based on the canonical construction path method of a given order. Its description can be found in Appendix D.1 and a rough outline of its recursive method is given in Algorithm 17.

Firstly, we verified that the implementations of Algorithm 7 and Algorithm 17 yield the same results. All counts we checked in this way were in agreement. For cycle permutation graphs this was checked up to order 34. For cycle permutation graphs of girth at least 5 up to order 34, of girth at least 6 up to order 38, of girth at least 7 up to order 42, of girth at least 8 up to order 50 and of girth at least 9 up to order 64. Variations of these algorithms allow us to generate non-hamiltonian cycle permutation graphs. We compared the results for non-hamiltonian cycle permutation graphs of girth at least 5 up to order 42, of girth at least 6 up to order 42, for girth at least 7 up to order 50, for girth at least 8 up to order 56, and for girth at least 9 up to order 68 and also here all counts were in agreement.

Secondly, we wrote a filter program, which given a cubic graph as input, determines whether it has a permutation 2-factor, by generating all perfect matchings of the graph, looking at their complement and verifying whether the corresponding 2-factor consists of two chordless cycles. Its implementation is open source and can be found in the same repository as our other programs on GitHub [52]. This can then be combined with a generator for cubic graphs in order to determine the counts of cycle permutation graphs. We used

`snarkhunter` [17, 19], which is a state-of-the-art generator for cubic graphs which allows to specify a lower bound on the girth.

Using this generator-filter approach we verified the counts of cycle permutation graphs up to order 26. For cycle permutation graphs with girth at least 5, we verified this up to order 30. For girth at least 6, we verified up to order 34 and for girth at least 7 we verified up to order 38. For cubic graphs of girth at least 8 or 9, we obtained these graphs from the meta-directory at the House of Graphs [30] at `https://houseofgraphs.org/meta-directory/cubic`. We then used the filter on these graphs and verified the counts up to order 46 and 64, respectively.

Another way we verified our counts is by using an algorithm that uses isomorphism-by-lists (i.e. the memory intensive approach mentioned in the first paragraph of Section 5.2) instead of the orderly generation or canonical construction path method. Using this method, we also verified the counts of cycle permutation graphs up to order 28, for girth at least 5 also up to order 28, for girth at least 6 up to order 32, for girth at least 7 up to order 36, for girth at least 8 up to 46 and for girth at least 9 up to 62.

We have also used alternative implementations to verify the counts of non-hamiltonian cycle permutation graphs. One of them uses isomorphism-by-lists and a backtracking approach for checking hamiltonicity which uses the implementation that can be found on GitHub in [56] and which was developed for the paper [53], instead of `cubhamg` (see Section 5.2.2). Using this approach we verified the counts of non-hamiltonian cycle permutation graphs up to order 34, with girth at least 6 up to order 38, with girth at least 7 up to order 44, with girth at least 8 up to order 52 and with girth at least 9 up to order 64.

A second implementation uses the canonical construction path method as well as this method for checking the hamiltonicity of the intermediate graphs. With this implementation we verified the counts of non-hamiltonian cycle permutation graphs up to order 40, with girth at least 6 up to order 36, with girth at least 7 up to order 44, with girth at least 8 up to order 52 and with girth at least 9 up to order 64. (Note that we allotted a lot more computation time to the generation of general non-hamiltonian cycle permutation graphs using this method, which is why we were able to verify this up to such a large order when compared to the generation of these graphs with girth restrictions.)

For all cases described above, the obtained results were in complete agreement, which gives a lot of confidence about the correctness of our implementation. Moreover, our implementations of these algorithms are open source software and can be found on GitHub [52] where they can be verified and used by other researchers.

D.3 Optimisations for determining canonicity

When determining whether or not a sequence p representing G via a permutation 2-factor F is canonical during a run of Algorithm 7, we use several optimisations to speed up this process.

We first note that, in practice, we work with the difference lists of the sequences representing the graphs rather than the sequences themselves. While this slightly changes whether or not a labelling is canonical, the resulting graphs output by the algorithm will still be the same. This representation makes it easier to look at a sequence and see the effect of its rotations.

We define the difference list of a sequence p, where k is the length of p, as

$$d(p) : [k] \to [k] \cup \{?\} : i \mapsto \begin{cases} ? & \text{if } p(i) = ? \text{ or } p(i+1) = ?, \\ p(i+1) - p(i) & \text{if } p(i) < p(i+1) \text{ or} \\ k + p(i+1) - p(i) & \text{otherwise,} \end{cases}$$

taking indices modulo k.

Suppose we encounter p in our algorithm and its first m entries are filled, i.e. not equal to ?, and the remaining entries are ?. We use the knowledge that $p|_{m-1}$ was canonical to speed up some of the computations of determining whether p is canonical.

Let $d := d(p)$ be the difference list of p and let F be the natural 2-factor obtained from the labelling $G(p)$. When computing whether or not d is canonical, we first check if it is weakly canonical before checking whether it is lexicographically smaller than sequences representing G via a different permutation 2-factor. In order to check its weak canonicity, we need to check the rotations of eight lists, namely the k rotations of d, the k rotations of d in the opposite direction, the k rotations of the list where we subtract each element of d from k, the k rotations of taking both the opposite direction and subtracting and the k rotations of the four previous lists, but for $d(I(p))$ instead of d. Note that $I(p)$ corresponds to swapping the cycles of F.

A naive approach would linearly check for each of these lists whether or not d is lexicographically smaller. However, for each of the lists we dynamically keep track which subsequences whose next element is ? are also a prefix of d. This allows us to often determine canonicity using only one comparison for each such subsequence. On average, there seem to be around 2 subsequences that match with d distributed over the 8 lists, so in practice the canonicity test is almost free. We illustrate this with an example.

Let $p = 0, 2, 6, 3, 5, 1, ?, ?$, then $d(p) = 2, 4, 5, 2, 4, ?, ?, ?$. Suppose we want to compare $d(p)$ to all of its rotations. Since the final 4 of $d(p)$ was a ? in the previous iteration (i.e. for $d(p|_5)$), we have stored "2" as a subsequence which is a prefix of $d(p)$. We have also stored the empty sequence as a subsequence. Since the empty sequence has length 0, we compare the first element of $d(p)$, i.e. 2, to the new 4. Since $2 < 4$ we cannot determine that $d(p)$ is not canonical. Since the sequence "2" has length 1, we compare the second element of $d(p)$, i.e. 4 to the new 4. If it were larger, then there is a rotation yielding a lexicographically smaller sequence than $d(p)$, if it were smaller, then this rotation is not lexicographically smaller and with the new element we cannot extend this subsequence to a new one which is still a prefix. Since in this case, the elements are equal and the 4 is proceeded by a ?, we extend the subsequence "2", to "2, 4" a new subsequence which is also a prefix of $d(p)$. We have now checked all rotations of $d(p)$ using only two comparisons. The same principle can be applied to the seven remaining lists we need to check. Using this optimisation to generate cycle permutation graphs using the weak orderly generation approach, the runtime decreases approximately by a factor of 3 for orders 26 and 28, when compared to an approach that linearly determines the rotation of the list which is lexicographically minimal and pruning as soon as we can decide it will be smaller than $d(p)$.

D.4 Runtimes and counts for the generation of cycle permutation graphs

Order	$g \geq 4$	$g \geq 5$	$g \geq 6$	$g \geq 7$	$g \geq 8$	$g \geq 9$
24	1.89s	$<$ 1s	$<$ 1s	$<$ 1s	$<$ 1s	$<$ 1s
26	30.16s	3.98s	$<$ 1s	$<$ 1s	$<$ 1s	$<$ 1s
28	533.48h	73.71*ss*	$<$ 1s	$<$ 1s	$<$ 1s	$<$ 1s
30	**3.93h**	1420.06s	6.70s	$<$ 1s	$<$ 1s	$<$ 1s
32	**82.86h**	**10.89h**	152.78s	$<$ 1s	$<$ 1s	$<$ 1s
34	**0.22yrs**	**248.10h**	**1.41h**	$<$ 1s	$<$ 1s	$<$ 1s
36	/	**0.68yrs**	**36.59h**	1.60s	$<$ 1s	$<$ 1s
38	/	/	**0.11yrs**	60.29s	$<$ 1s	$<$ 1s
40	/	/	/	2702.68s	$<$ 1s	$<$ 1s
42	/	/	/	**40.19h**	$<$ 1s	$<$ 1s
44	/	/	/	**0.19yrs**	$<$ 1s	$<$ 1s
46	/	/	/	/	4.00s	$<$ 1s
48	/	/	/	/	257.71s	$<$ 1s
50	/	/	/	/	**6.09h**	$<$ 1s
52	/	/	/	/	**455.49h**	$<$ 1s
54	/	/	/	/	/	$<$ 1s
56	/	/	/	/	/	$<$ 1s
58	/	/	/	/	/	$<$ 1s
60	/	/	/	/	/	1.57s
62	/	/	/	/	/	70.28s
64	/	/	/	/	/	1.30h
66	/	/	/	/	/	**203.94h**

Table D.1: Runtimes for computing the entries in Table 5.1 using the (strong) orderly generation method. Bold entries were performed by parallelising the computation (which introduces some overhead) and executing it on the supercomputer of the Flemish supercomputer center (VSC), using Intel Xeon Platinum 8360Y (IceLake) CPUs.

Order	Alg. 7 counts	Alg. 8 counts	Rel. (%)	Time Alg. 7	Time Alg. 8	Speedup
6	1	1	100.00	< 1s	< 1s	/
8	2	2	100.00	< 1s	< 1s	/
10	4	4	100.00	< 1s	< 1s	/
12	10	10	100.00	< 1s	< 1s	/
14	28	28	100.00	< 1s	< 1s	/
16	123	127	96.85	< 1s	< 1s	/
18	667	686	97.23	< 1s	< 1s	/
20	4 815	4 975	96.78	< 1s	< 1s	/
22	41 369	42 529	97.27	< 1s	< 1s	/
24	411 231	420 948	97.69	1.89s	0.45s	4.20
26	4 535 796	4 622 509	98.12	30.16s	4.79s	6.30
28	54 828 142	55 670 332	98.49	533.48s	62.07s	8.59
30	717 967 102	726 738 971	98.79	**3.93h**	861.05s	16.43
32	10 118 035 593	10 217 376 792	99.03	**82.86h**	**5.02h**	16.50
34	152 626 831 184	153 848 448 652	99.21	**0.22yrs**	**95.86h**	19.81
36	?	2 470 073 249 960	/	/	**0.15yrs**	/

Table D.2: Counts of cycle permutation graphs output by the implementations of Algorithms 7 and 8 using the strong and weak orderly generation methods (resp. second and third column). The fourth column displays the percentage of non-isomorphic graphs present in the third column. The fifth and sixth column display the running times to obtain these counts of Algorithm 7 and 8, respectively. The final column displays by what factor the sixth column is faster than the fifth. Bold entries were performed by parallelising the computation (which introduces some overhead) and executing it on the Flemish supercomputer center (VSC), using Intel Xeon Platinum 8360Y (IceLake) CPUs.

D.5 Runtimes and counts for the generation of non-hamiltonian cycle permutation graphs and permutation snarks

Order	$g \geq 5$	$\chi' = 4$	$g \geq 6$	$g \geq 7$	$g \geq 8$	$g \geq 9$
30	3.08s	< 1s	< 1s	< 1s	< 1s	< 1s
32	25.21s	4.43s	< 1s	< 1s	< 1s	< 1s
34	233.91s	40.11s	3.98s	< 1s	< 1s	< 1s
36	**1.01h**	275.93s	35.34s	< 1s	< 1s	< 1s
38	**8.52h**	**1.48h**	**0.33h**	< 1s	< 1s	< 1s
40	**77.89h**	**10.91h**	**1.93h**	2.83s	< 1s	< 1s
42	**775.81h**	**115.22h**	**16.13h**	28.49s	< 1s	< 1s
44	**0.75yrs**	**0.11yrs**	**132.93h**	316.74s	< 1s	< 1s
46	**8.16yrs**	**1.30yrs**	**0.17yrs**	**1.78h**	< 1s	< 1s
48	/	/	**1.92yrs**	**15.73h**	1.16s	< 1s
50	/	/	/	**174.16h**	14.83s	< 1s
52	/	/	/	**0.24yrs**	236.57s	< 1s
54	/	/	/	**2.94yrs**	**1.66h**	< 1s
56	/	/	/	/	**21.86h**	< 1s
58	/	/	/	/	**317.83h**	< 1s
60	/	/	/	/	**0.62yrs**	0.73s
62	/	/	/	/	/	12.21s
64	/	/	/	/	/	258.18s
66	/	/	/	/	/	**2.81h**
68	/	/	/	/	/	**54.14h**
70	/	/	/	/	/	**0.12yrs**

Table D.3: Runtimes for computing the entries in Table 5.2 using Algorithm 9 (and Algorithm 10 for the third column) based on weak orderly generation. Bold entries were performed by parallelising the computation (which introduces some overhead) and executing it on the Flemish supercomputer center (VSC), using Intel Xeon Platinum 8360Y (IceLake) CPUs.

Order	$g \geq 5$	$g \geq 6$	$g \geq 7$	$g \geq 8$	$g \geq 9$
26	3.18s	$<$ 1s	$<$ 1s	$<$ 1s	$<$ 1s
28	26.76s	0.49s	$<$ 1s	$<$ 1s	$<$ 1s
30	261.91s	4.59s	$<$ 1s	$<$ 1s	$<$ 1s
32	**1.43h**	43.99s	$<$ 1s	$<$ 1s	$<$ 1s
34	**9.84h**	429.50s	$<$ 1s	$<$ 1s	$<$ 1s
36	**105.20h**	**3.61h**	1.71s	$<$ 1s	$<$ 1s
38	**1062.77h**	**26.53h**	19.10s	$<$ 1s	$<$ 1s
40	**1.41yrs**	**251.97h**	224.90s	$<$ 1s	$<$ 1s
42	**17.66yrs**	**3412.92h**	**2.89h**	$<$ 1s	$<$ 1s
44	/	/	**20.98h**	0.50s	$<$ 1s
46	/	/	**261.32h**	6.02s	$<$ 1s
48	/	/	**3041.08h**	89.93s	$<$ 1s
50	/	/	**5.24yrs**	1480.43s	$<$ 1s
52	/	/	/	**13.26h**	$<$ 1s
54	/	/	/	**204.36h**	$<$ 1s
56	/	/	/	**3448.13h**	0.15s
58	/	/	/	/	2.17s
60	/	/	/	/	39.36s
62	/	/	/	/	838.34s
64	/	/	/	/	**11.60h**
66	/	/	/	/	**275.49h**
68	/	/	/	/	**6185.93h**

Table D.4: Runtimes for computing the entries in Table 5.2 using Algorithm 17 based on the canonical construction path method. Bold entries were performed by parallelising the computation (which introduces some overhead) and executing it on the Flemish supercomputer center (VSC), using Intel Xeon Platinum 8360Y (IceLake) CPUs.

Order	Non-ham. counts	Alg. 9 counts	Rel. (%)	Snark counts	Alg. 10 counts	Rel. (%)
10	1	1	100.00	1	1	100.00
12-16	0	0	100.00	0	0	100.00
18	2	4	50.00	2	4	50.00
20	0	0	100.00	0	0	100.00
22	1	1	100.00	0	0	100.00
24	0	0	100.00	0	0	100.00
26	64	174	36.78	64	174	36.78
28	0	0	100.00	0	0	100.00
30	9	18	50.00	0	0	100.00
32	0	0	100.00	0	0	100.00
34	10 778	33 790	31.90	10 771	33 767	31.90
36	4	23	17.39	0	0	100.00
38	1 848	5 124	36.07	0	0	100.00
40	19	196	9.69	0	0	100.00
42	3 131 740	10 789 929	29.02	3 128 893	10 774 396	29.04
44	1 428	16 640	8.58	0	0	100.00
46	678 106	2 327 096	29.14	0	0	100.00

Table D.5: The number of non-hamiltonian cycle permutation graphs (permutation snarks) output by the implementation of Algorithm 8 using the weak orderly generation method. These counts are found in the third (sixth) column. The second (fifth) column shows the total number of pairwise non-isomorphic non-hamiltonian cycle permutation graphs (permutation snarks) for each order.

Appendix E

Appendix to Chapter 6

E.1 Correctness tests

We performed various tests for verifying the correctness of the implementation of our algorithm described in Section 6.2 for counting the number of HISTs and testing HIST-criticality. Our implementation can be found on GitHub [50]. We will call this Implementation A. An independent implementation of the algorithm which varies only slightly in the way edges are chosen to be added to the subtree was written by Andreas Awouters. We will call this Implementation B.

We also implemented another branch and bound algorithm which counts the number of HISTs in a graph based on an algorithm for the generation of spanning trees by Kapoor and Ramesh [86]. We call this Implementation C. It starts by creating an initial spanning tree T of the graph G using depth first search. This does not need to be a HIST, but to have the correct counts, we check whether it is one. We apply the recursive algorithm to such a tree T, where some of the edges of G are marked with "in" or with "out". "In" meaning it will remain in the trees generated by this branch of the search tree. "Out" meaning it will never belong to the trees generated by this branch of the search tree.

For a call of the recursive algorithm, we choose a non-edge e of T which has not been marked as "out" and compute its *fundamental cycle* in T, i.e. the unique cycle containing e in $T + e$. Denote its edges by $e, f_1, \ldots, f_k$. Let f_i be the first edge not marked as "in". We mark e as "in" and f_i as "out" and recursively apply the algorithm to $T + e - f_i$. After this, we mark f_i as "in" and mark the next edge f_j not marked "in" as "out" and apply the algorithm recursively to $T + e - f_j$. Then we also mark f_j as "in", etc. until we have done this for all

edges not marked “in" of the fundamental cycle. Finally, we unmark the edges we just marked as “in" and mark e as “out" and apply the recursive algorithm to T.

Because the edges marked “in" and “out" lay restrictions on the edges present in the spanning trees we generate in branches of the search tree, we can apply similar pruning rules as described in the algorithm of Section 6.2. For example when a vertex v has two incident edges marked “in" and all other incident edges marked “out", then we can already backtrack, since all trees which will be generated in this branch of the search space will have v as a degree 2 vertex.

First of all, since all algorithms are branch and bound algorithms based on algorithms for generating spanning trees, we can remove the pruning criteria specifically for HISTs and see if they correctly count the number of spanning trees of graphs. We verified this for all algorithms for a large sample of graphs, whose number of spanning trees can easily be computed using Kirchhoff's Matrix Tree Theorem.

We used Implementations B and C to verify the counts of Table 6.1 obtained by Implementation A. Since Implementations B and C are a bit slower than Implementation A, we were not able to double-check all counts, but we verified the following.

In the general case both B and C verified the counts up to and including order 11, for girth at least 4 both implementations verified the counts up to order 14, for girth at least 5 both implementations verified the counts up to and including order 17, for girth at least 6 both implementations verified the counts up to and including order 19 and for girth at least 7 both implementations verified the counts up to and including order 21.

We also verified the counts of Table 6.2 obtained by Implementation A. Both Implementations B and C verified the counts up to and including order 13.

Implementations B and C verified the counts of Table 6.3 up to and including order 14, hence also verifying the results of Proposition 4 up to this order.

E.2 Smallest HIST-critical graphs

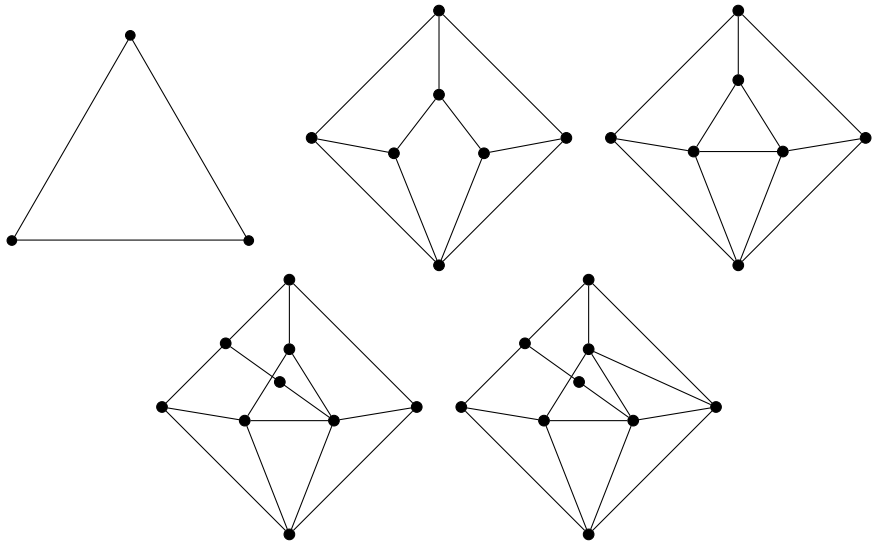


Figure E.1: The five HIST-critical graphs with smallest order.

E.3 Drawings of HIST-critical graphs with a specific girth

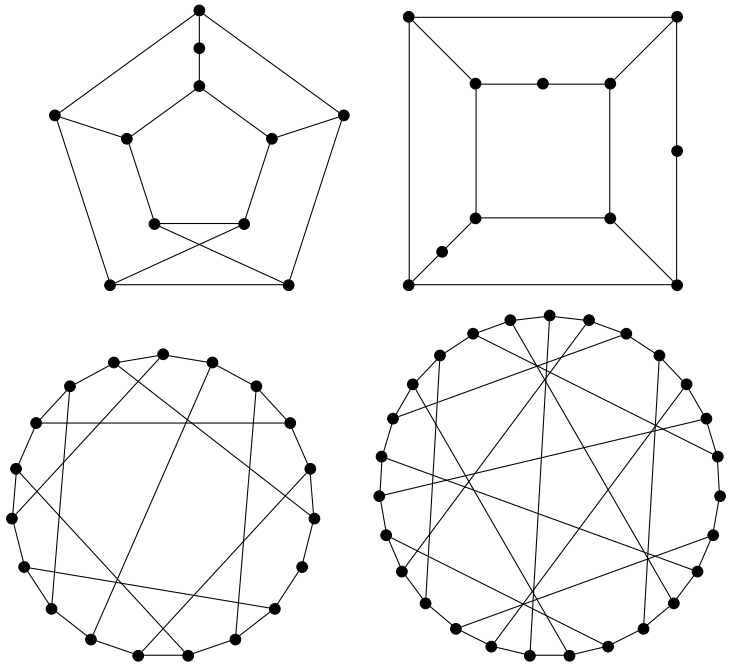


Figure E.2: Smallest HIST-critical graphs of girth $4, 5, 6$ and 7, respectively.

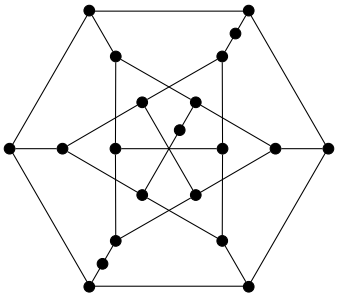
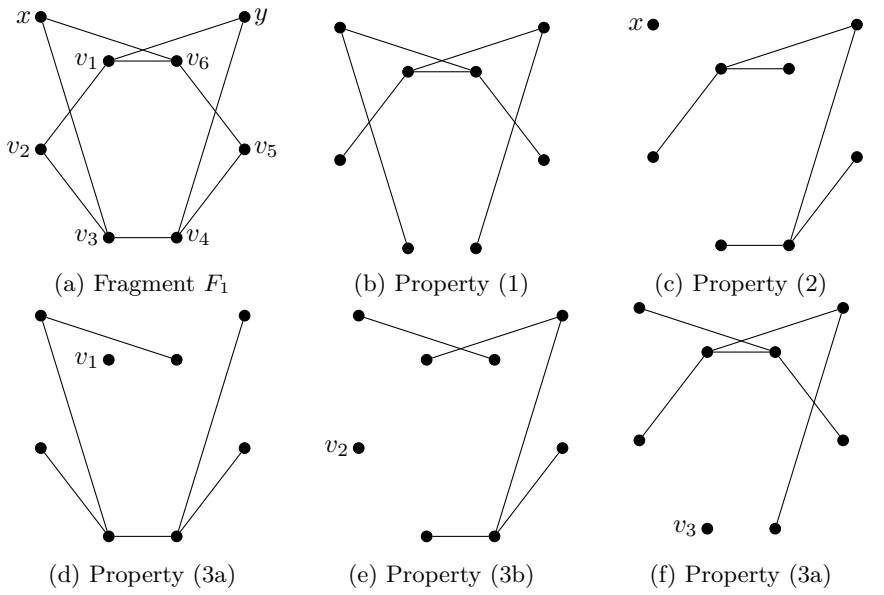


Figure E.3: A HIST-critical graph of girth 6. It is the Pappus graph with three edges subdivided.

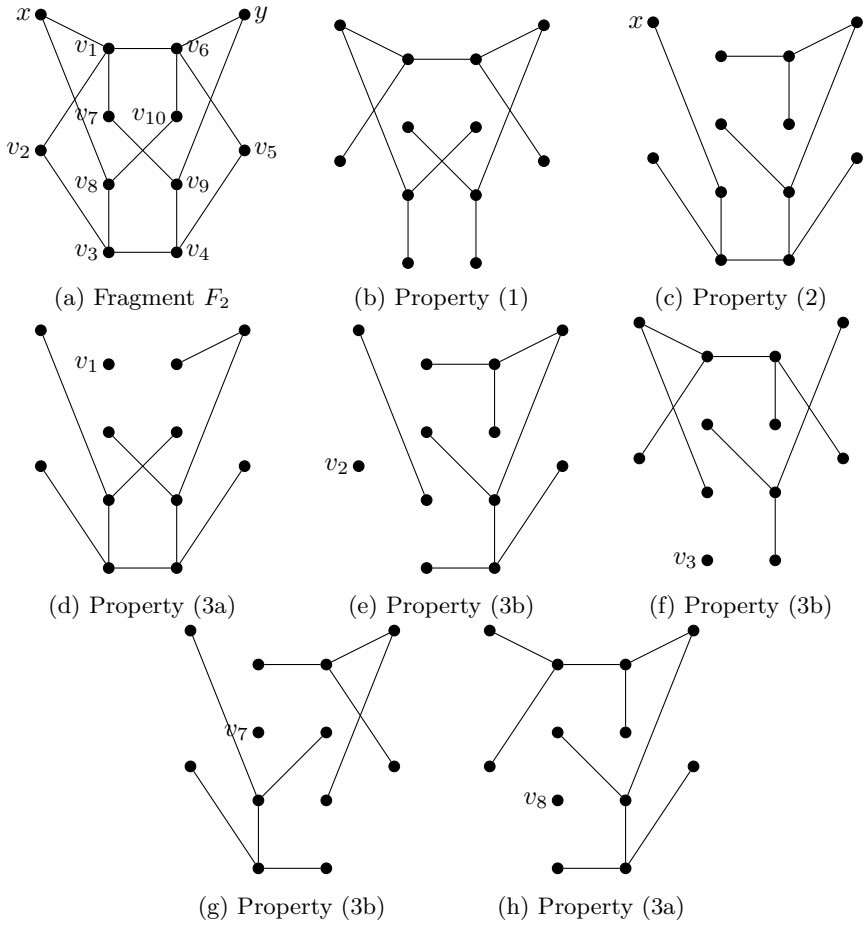
E.4 Certificates for the proof of Proposition 6.3.



(a) Fragment F_1 (b) Property (1) (c) Property (2)

(d) Property (3a) (e) Property (3b) (f) Property (3a)

Figure E.4: Fragment F_1 with properties (2) and (3) defined in Section 3.1.



(a) Fragment F_2 (b) Property (1) (c) Property (2)

(d) Property (3a) (e) Property (3b) (f) Property (3b)

(g) Property (3b) (h) Property (3a)

Figure E.5: Fragment F_2 with properties (2) and (3) defined in Section 3.1.

E.5 Number of planar 4-connected graphs

Order	# 4-conn. planar
6	1
7	1
8	4
9	10
10	53
11	292
12	2 224
13	18 493
14	167 504
15	1 571 020
16	15 151 289
17	148 864 939
18	1 485 904 672
19	15 028 654 628
20	153 781 899 708
21	1 589 921 572 902
22	16 591 187 039 082

Table E.1: The number of planar 4-connected graphs for each order.

E.6 Planar 4-connected graphs with fewest number of HISTs

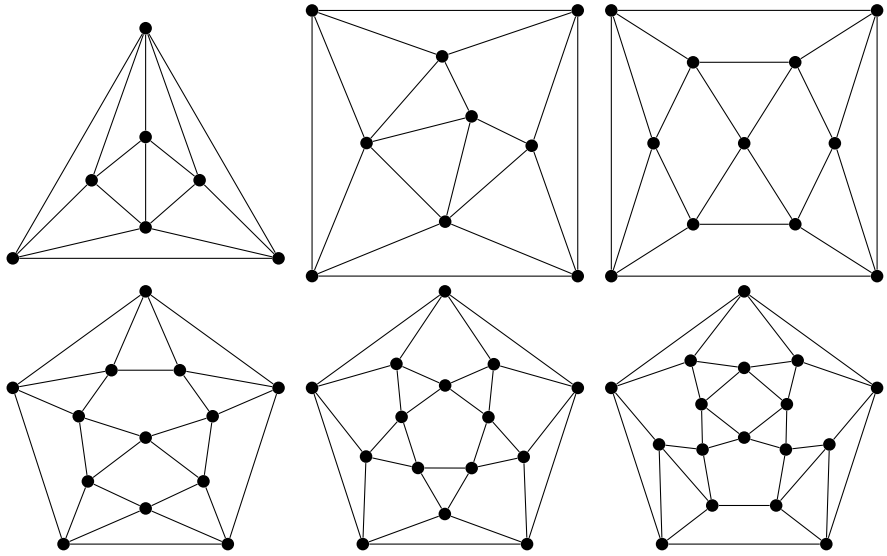


Figure E.6: The planar 4-connected graphs of odd order n attaining the minimum number of HISTs for each order up to order 17.

Appendix F

Appendix to Chapter 7

F.1 Figure in the proof of Theorem 7.2

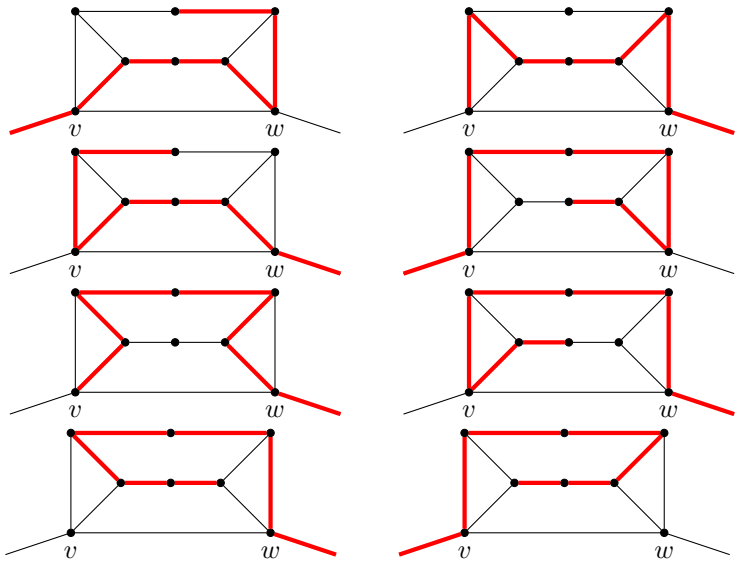


Figure F.1: Hamiltonian paths in vertex-deleted subgraphs of Ξ_8.

F.2 Example illustrating the fault cost definition

Consider the graph Ξ_9 shown in Figure F.2. It is not difficult to check that Ξ_9 is the smallest 2-leaf-guaranteed graph, both in terms of order and size. As a 2-leaf-guaranteed graph, Ξ_9 is non-hamiltonian but traceable, i.e. its minimum leaf number is 2, and all of its vertex-deleted subgraphs are traceable.

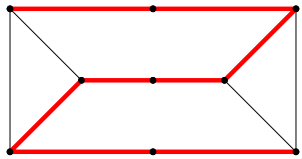


Figure F.2: The graph Ξ_9. Its minimum leaf number is 2. In bold red an ml-subgraph S in Ξ_9 is shown.

Denote the trees given in the first row of Figure F.3 with S_1, S_2, S_3, S_4. We have $\tau(S, S_1) = 1$, $\tau(S, S_2) = \tau(S, S_3) = \tau(S, S_4) = 3$. Therefore, if v_1 fails, the transition cost from S to an ml-subgraph in $\Xi_9 - v_1$ is at least 1. If v_2 fails, the transition cost from S to an ml-subgraph in $\Xi_9 - v_2$ is 4, and for every $i \in \{3, 4, 5\}$, if v_i fails, the transition cost from S to an ml-subgraph in $\Xi_9 - v_i$ is 2. By symmetry, this covers all cases. Thus, for the ml-subgraph S specified in Figure F.2, we have

$$\max_{v \in V(\Xi_9)} \min_{S_v \in \mathcal{S}_{\mathrm{ml}}(\Xi_9 - v)} \tau(S, S_v) = 4.$$

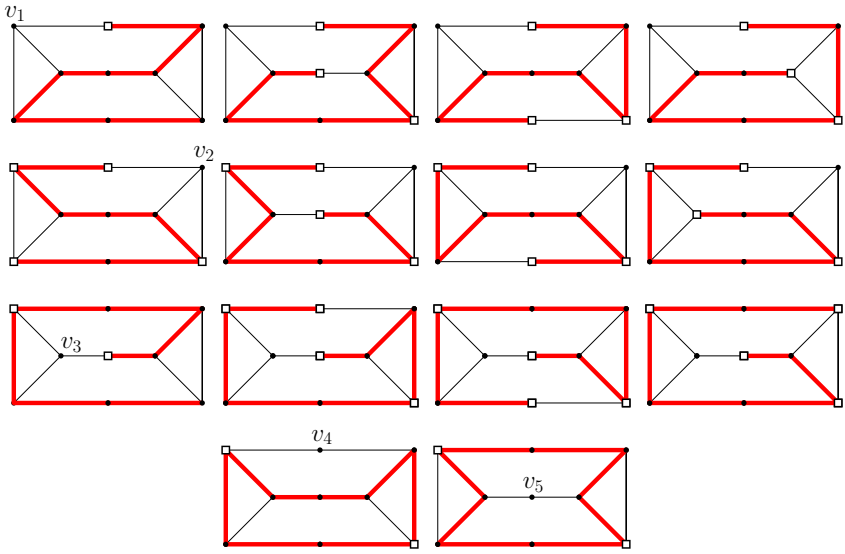


Figure F.3: Ignoring symmetric cases, the above shows all ml-subgraphs S_v in $\Xi_9 - v$: twelve cases if $v \in \{v_1, v_2, v_3\}$ and two cases if $v \in \{v_4, v_5\}$. A vertex $w \neq v$ is depicted as a white square whenever $\deg_S(w) \neq \deg_{S_v}(w)$, so the number of such vertices is $\tau(S, S_v)$.

Performing this straightforward analysis for all other ml-subgraphs of Ξ_9, we obtain that $\varphi(\Xi_9) = 2$. An optimal subgraph in Ξ_9, i.e. an ml-subgraph realising this minimum fault cost, is shown in Figure F.4, left-hand side. Finally, we point out that Ξ_9 does contain an ml-subgraph S and a vertex v such that $\Xi_9 - v$ contains an ml-subgraph S_v with $\tau(S, S_v) = 0$, see Figure F.4. Graphs with vanishing fault cost are characterised in Proposition 7.3.

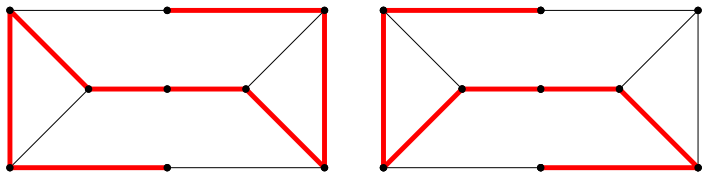


Figure F.4: An ml-subgraph of Ξ_9 (left-hand side) and an ml-subgraph of a vertex-deleted subgraph of Ξ_9 (right-hand side). The two subgraphs shown in this figure have transition cost 0.

F.3 Correctness tests

While it is relatively easy to prove the correctness of Algorithm 11, we also performed various tests to verify the correctness of the implementation. Our implementation of the algorithm is open source software and can be found on GitHub [55], where it can be verified and used by other researchers.

First of all, we verified that the examples of Corollary 7.8, indeed have fault cost 3 up to $k = 5$. Our program determined that this was indeed the case.

Next, we verified for the family of Theorem 7.3 that the fault costs obtained by our program are the same as what is stated in the proof of the theorem. We checked this for both constructions up to order 12.

We also verified with our program that the two examples of Figure 7.8 indeed have fault cost 1.

Algorithm 11 can easily be adapted to determine the minimum leaf number of the given graphs. We compared the output of our program for the minimum leaf number to a second independent implementation of an algorithm for determining the minimum leaf number by Floor Van de Steene [135]. Both programs were in agreement for all of our checks. In particular, we checked 2-connected graphs up to order 9, 2-connected graphs of girth at least 4 up to order 11, 2-connected graphs of girth at least 5 up to order 15, 3-connected graphs of girth at least 4 up to order 12, 3-connected graphs of girth at least 5 up to order 16, 2-connected cubic graphs up to order 18, 2-connected cubic graphs of girth at least 4 up to order 20, 2-connected cubic graphs of girth at least 5 up to order 22 and 3-connected planar graphs up to order 10.

By Proposition 7.3 all 1-hamiltonian graphs have fault cost 0. Previously, we had already implemented an algorithm which can determine if a graph is 1-hamiltonian (the source code of this program can be found on GitHub [56]). We used this program to filter the 1-hamiltonian ones in various classes of graphs and then we checked for these graphs whether our program correctly detects that their fault cost is 1. Everything we checked was in agreement. In particular, we checked 2-connected graphs up to order 11, 2-connected graphs of girth at least 4 up to order 13, 2-connected graphs of girth at least 5 up to order 16, 2-connected cubic graphs up to order 22, cubic graphs of girth at least 5 up to order 24, and 3-connected planar graphs up to order 12.

F.4 Graphs realising the minimum of Proposition 7.4

In Proposition 7.4 we provided the order of the smallest graphs attaining fault cost k for $k \leq 8$, with the exception of $k = 1$. For every k, the most symmetric graphs can be found on the House of Graphs [30] by searching for the keywords "fault cost". We give an example of each in Figure F.5.

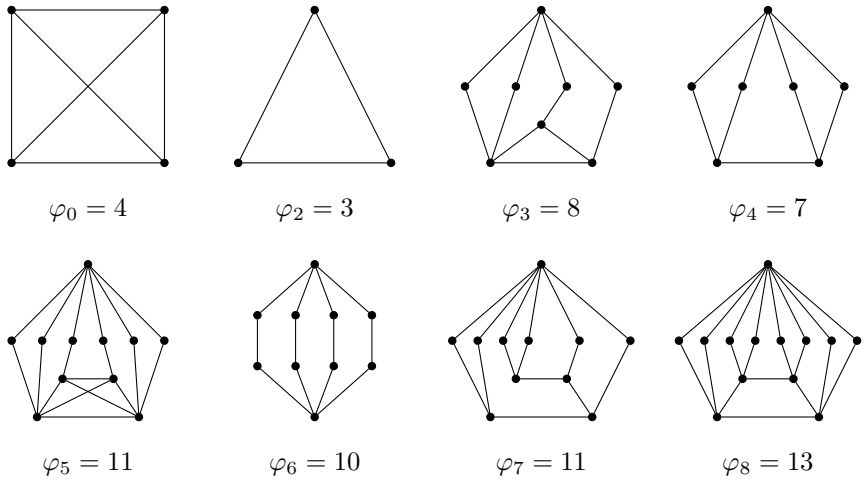


Figure F.5: A smallest graph attaining fault cost k for $k \leq 8$, with the exception of $k = 1$ (we do not know the exact value of φ_1). See Proposition 7.4. Each of these graphs is one with largest automorphism group size out of all graphs attaining these values with the exception of $k = 8$, where this is only true for the graphs of girth at least 4.

F.5 Fault costs of 3-connected graphs

$n\backslash\varphi$	0	1	2	3	4
4	1	0	0	0	0
5	2	0	0	0	0
6	9	0	4	0	0
7	91	0	20	0	0
8	1 636	0	368	0	0
9	58 119	0	8 291	0	0
10	3 575 889	0	326 455	0	0
11	369 791 302	0	18 832 723	0	81
12	63 452 885 511	0	1 689 918 189	0	1 040
13	256 052	0	612 280	0	330
14	11 309 365	0	17 621 062	215	1 536
15	62	0	87	0	0
16	984	0	686	0	0
17	16 590	0	7 292	0	0
18	327 612	0	94 582	0	0
19	7 213 982	0	1 330 513	0	0
20	173 890 208	0	21 401 341	0	0

Table F.1: Counts of how many 3-connected graphs attain each fault cost φ for each order n. The top part of the table gives counts for all 3-connected graphs, the middle part for 3-connected graphs of girth at least 4, and the bottom part for 3-connected graphs of girth at least 5. Fault costs for which the count is zero or which are not included in the table imply that no graphs of the given orders attain this fault cost.

F.6 Fault costs of 3-connected cubic graphs

$n\backslash\varphi$	0	1	2
4	1	0	0
6	1	0	1
8	2	0	2
10	6	0	8
12	27	0	30
14	158	0	183
16	1 396	0	1 432
18	16 067	0	14 401
20	227 733	0	168 417
22	3 740 294	0	2 168 998
24	68 237 410	0	29 863 609
26	1 346 345 025	0	436 047 621
28	7 352 343 711	0	1 103 915 037
30	14 468 621 439	0	152 588 083

Table F.2: Counts of how many 3-connected cubic graphs attain each fault cost φ for each order n. The top part of the table gives counts for all 3-connected graphs cubic , the middle part for 3-connected cubic graphs of girth at least 4, and the bottom part for 3-connected cubic graphs of girth at least 5. Fault costs for which the count is zero or which are not included in the table imply that no graphs of the given orders attain this fault cost.

F.7 Fault costs of 3-connected planar graphs

$n\backslash\varphi$	0	1	2
4	1	0	0
5	2	0	0
6	7	0	0
7	34	0	0
8	246	0	11
9	2 526	0	80
10	30 842	0	1 458
11	416 108	0	24 456
12	5 955 716	0	428 918
13	88 766 610	0	7 496 328
14	1 364 787 597	0	131 437 755
15	21 522 536 388	0	2 311 451 741
16	346 748 111 059	0	40 843 399 185

Table F.3: Counts of fault costs for 3-connected planar graphs. Fault costs for which the count is zero or which are not included in the table imply that no graphs of the given orders attain this fault cost.

Bibliography

[1] M. O. Albertson, D. M. Berman, J. P. Hutchinson and C. Thomassen. Graphs with homeomorphically irreducible spanning trees. *J. Graph Theory*, 14:247–258, 1990. DOI: 10.1002/jgt.3190140212.

[2] R. E. L. Aldred, B. D. McKay and N. C. Wormald. Small hypohamiltonian graphs. *J. Comb. Math. Comb. Comput.*, 23:143–152, 1997.

[3] N. Alon and M. Tarsi. Covering multigraphs by simple circuits. *SIAM J. Alg. Discret. Meth.*, 6(3):345–350, 1985. DOI: 10.1137/0606035.

[4] K. Appel and W. Haken. Every planar map is four colorable. Part I: Discharging. *Ill. J. Math.*, 21(3):429–490, 1977. DOI: 10.1215/ijm/1256049011.

[5] K. Appel, W. Haken and J. Koch. Every planar map is four colorable. Part II: Reducibility. *Ill. J. Math.*, 21(3):491–567, 1977. DOI: 10.1215/ijm/1256049012.

[6] K. Appel and W. Haken. *Every Planar Map is Four Colorable*, volume 98 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, USA, 1989. DOI: 10.1090/conm/098.

[7] M. Araya and G. Wiener. On cubic planar hypohamiltonian and hypotraceable graphs. *Electron. J. Comb.*, 18(1):P85, 2011. DOI: 10.37236/572.

[8] L. Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, STOC '16, pages 684–697, New York, NY, USA. Association for Computing Machinery, 2016. DOI: 10.1145/2897518.2897542.

[9] A. T. Balaban. Mathematical chemistry: (3,g)-cages with girth g, topological indices, and other graph-theoretical problems. *Fundam. Inform.*, 64(1-4):1–16, 2005.

[10] J. Barát and Z. L. Blázsik. Improved upper bound on the Frank number of 3-edge-connected graphs. *Eur. J. Comb.*, 118:103913, 2024. DOI: 10.1016/j.ejc.2023.103913.

[11] J. Barát and Z. L. Blázsik. Quest for graphs of Frank number 3. *Australas. J. Comb.*, 88:52–76, 2024.

[12] D. Binkele-Raible, H. Fernau, S. Gaspers and M. Liedloff. Exact and parameterized algorithms for max internal spanning tree. *Algorithmica*, 65:95–128, 2013. DOI: 10.1007/s00453-011-9575-5.

[13] J. A. Bondy. Variations on the hamiltonian theme. *Canad. Math. Bull.*, 15(1):57–62, 1972. DOI: 10.4153/CMB-1972-012-3.

[14] J. A. Bondy and V. Chvátal. A method in graph theory. *Discret. Math.*, 15(2):111–135, 1976. DOI: 10.1016/0012-365X(76)90078-9.

[15] J. M. Boyer and W. J. Myrvold. On the cutting edge: simplified $O(n)$ planarity by edge addition. *J. Graph Algorithms Appl.*, 8(3):241–273, 2004. DOI: 10.1142/9789812773289_0014.

[16] G. Brinkmann. Isomorphism rejection in structure generation programs. In *Discrete Mathematical Chemistry*, volume 51 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2000. DOI: 10.1090/dimacs/051.

[17] G. Brinkmann and J. Goedgebeur. Generation of cubic graphs and snarks with large girth. *J. Graph Theory*, 86(2):255–272, 2017. DOI: 10.1002/jgt.22125.

[18] G. Brinkmann, J. Goedgebeur, J. Hägglund and K. Markström. Generation and properties of snarks. *J. Comb. Theory, Ser. B*, 103(4):468–488, 2013. DOI: 10.1016/j.jctb.2013.05.001.

[19] G. Brinkmann, J. Goedgebeur and B. D. McKay. Generation of cubic graphs. *Discret. Math. Theor. Comput. Sci.*, 13(2):69–80, 2011. DOI: 10.46298/dmtcs.551.

[20] G. Brinkmann, J. Goedgebeur and B. D. McKay. The minimality of the Georges–Kelmans graph. *Math. Comp.*, 91:1483–1500, 2022. DOI: 10.1090/mcom/3701.

[21] G. Brinkmann and B. D. McKay. Fast generation of planar graphs. *MATCH Commun. Math. Comput. Chem.*, 58(2):323–357, 2007.

[22] G. Brinkmann and B. D. McKay. Construction of planar triangulations with minimum degree 5. *Discret. Math.*, 301(2–3):147–163, 2005. DOI: 10.1016/j.disc.2005.06.019.

[23] G. Brinkmann, S. Greenberg, C. Greenhill, B. D. McKay, R. Thomas and P. Wollan. Generation of simple quadrangulations of the sphere. *Discret. Math.*, 305(1–3):33–54, 2005. DOI: 10.1016/j.disc.2005.10.005.

[24] G. Chartrand, S. F. Kapoor and D. R. Lick. n-hamiltonian graphs. *J. Comb. Theory, Ser. B.*, 9(3):308–312, 1970. DOI: 10.1016/S0021-9800(70)80069-2.

[25] G. Chartrand and F. Harary. Planar permutation graphs. *Ann. Inst. H. Poincaré*, 3(4):433–438, 1967.

[26] G. Chen, H. Ren and S. Shan. Homeomorphically irreducible spanning trees in locally connected graphs. *Comb. Prob. Comput.*, 21(1–2):107–111, 2012. DOI: 10.1017/S0963548311000526.

[27] G. Chen and S. Shan. Homeomorphically irreducible spanning trees. *J. Comb. Theory, Ser. B*, 103(4):409–414, 2013. DOI: 10.1016/j.jctb.2013.04.001.

[28] V. Chvátal. Flip-flops in hypohamiltonian graphs. *Canad. Math. Bull.*, 16(1):33–41, 1973. DOI: 10.4153/CMB-1973-008-9.

[29] J. B. Collier and E. F. Schmeichel. New flip-flop constructions for hypohamiltonian graphs. *Discret. Math.*, 18(3):265–271, 1977. DOI: 10.1016/0012-365X(77)90130-3.

[30] K. Coolsaet, S. D'hondt and J. Goedgebeur. House of Graphs 2.0: a database of interesting graphs and more. *Discret. Appl. Math.*, 325:97–107, 2023. DOI: 10.1016/j.dam.2022.10.013. Available at https://houseofgraphs.org.

[31] A. Dehghan, K. Siuta, A. Skorupka, A. Dubey, A. Betlen, D. Miller, W. Xu, B. Kamiński and P. Prałat. Detecting bots in social-networks using node and structural embeddings. *J. Big Data*, 10(1):119, 2023. DOI: 10.1186/s40537-023-00796-3.

[32] A. Demers and A. Downing. Minimum leaf spanning tree. Patent number 6,105,018. Date of patent 15 Aug. 2000. Oracle Corporation, Redwood Shores, CA, USA. URL: https://patents.google.com/patent/US6105018A/en.

[33] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, Germany, 6th edition, 2025. DOI: 10.1007/978-3-662-70107-2.

[34] G. A. Dirac. Some theorems on abstract graphs. *Proc. Lond. Math. Soc.*, s3-2(1):69–81, 1952. DOI: 10.1112/plms/s3-2.1.69.

[35] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Comment. Acad. Sci. Petropol.*, 8:128–140, 1741. In Latin.

[36] G. Exoo and R. Jajcay. Dynamic cage survey. *Electron. J. Comb.* DOI: 10.37236/37. DS16: Jul 26–2013.

[37] I. A. Faradžev. Generation of nonisomorphic graphs with a given degree sequence. In *Algorithmic Studies in Combinatorics*, pages 11–19. Nauka, Moscow, Russia, 1978. In Russian.

[38] G. B. Faulkner and D. H. Younger. Non-hamiltonian cubic planar maps. *Discret. Math.*, 7(1–2):67–74, 1974. DOI: 10.1016/S0012-365X(74)80019-1.

[39] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Math. Program.*, 1:168–194, 1971. DOI: 10.1007/BF01584085.

[40] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, USA, 1st edition, 1979.

[41] L. Gargano, M. Hammar, P. Hell, L. Stacho and U. Vaccaro. Spanning spiders and light-splitting switches. *Discret. Math.*, 285(1–3):83–95, 2004. DOI: 10.1016/j.disc.2004.04.005.

[42] L. A. Goddyn, J. van den Heuvel and S. McGuinness. Removable circuits in multigraphs. *J. Comb. Theory, Ser. B*, 71(2):130–143, 1997. DOI: 10.1006/jctb.1997.1775.

[43] J. Goedgebeur, E. Máčajová and J. Renders. Frank-Number [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/Frank-Number.

[44] J. Goedgebeur, E. Máčajová and J. Renders. On the frank number and nowhere-zero flows on graphs. In D. Paulusma and B. Ries, editors, *Proceedings of the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG2023)* (Fribourg, Switzerland), volume 14093 of *Lecture Notes in Computer Science*, pages 363–375. Springer, Cham., 2023. DOI: 10.1007/978-3-031-43380-1_26.

[45] J. Goedgebeur, E. Máčajová and J. Renders. On the frank number and nowhere-zero flows on graphs. In D. Paulusma and B. Ries, editors, *Proceedings of the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG2023)* (Fribourg, Switzerland), volume 14093 of *Lecture Notes in Computer Science*, pages 363–375. Springer, Cham., 2023. DOI: 10.1007/978-3-031-43380-1_26.

[46] J. Goedgebeur, E. Máčajová and M. Škoviera. Smallest snarks with oddness 4 and cyclic connectivity 4 have order 44. *Ars Math. Contemp.*, 16(2):277–298, 2019. DOI: 10.26493/1855-3974.1601.e75.

[47] J. Goedgebeur, E. Máčajová and M. Škoviera. The smallest nontrivial snarks of oddness 4. *Discret. Appl. Math.*, 277:139–162, 2020. DOI: 10.1016/j.dam.2019.09.020.

[48] J. Goedgebeur, B. Meersman and C. T. Zamfirescu. Graphs with few hamiltonian cycles. *Math. Comp.*, 89:965–991, 2020. DOI: 10.1090/mcom/3465.

[49] J. Goedgebeur, K. Noguchi, J. Renders and C. T. Zamfirescu. HIST-critical graphs and Malkevitch's conjecture. *arXiv preprint*, 2024. arXiv: 2401.04554.

[50] J. Goedgebeur, K. Noguchi, J. Renders and C. T. Zamfirescu. HistChecker [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/HistChecker.

[51] J. Goedgebeur, K. Ozeki, N. Van Cleemput and G. Wiener. On the minimum leaf number of cubic graphs. *Discret. Math.*, 342(11):3000–3005, 2019. DOI: 10.1016/j.disc.2019.06.005.

[52] J. Goedgebeur, J. Renders and S. Van Overberghe. Cycle Permutation Graphs [Computer software], version 2, 2025. URL: https://github.com/JarneRenders/Cycle-Permutation-Graphs.

[53] J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. K_2-hamiltonian graphs: II. *J. Graph Theory*, 105(4):580–611, 2024. DOI: 10.1002/jgt.23057.

[54] J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. Fault-tolerance of leaf-guaranteed graphs. In T. Jordán, G. Katona, C. Király and G. Wiener, editors, *Proceedings of the 12th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, Budapest, Hungary*, pages 585–591, 2023.

[55] J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. FaultCost [Computer software], version 1, 2025. URL: https://github.com/JarneRenders/faultCost.

[56] J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. K2-Hamiltonian Graphs [Computer software], version 1, 2022. URL: https://github.com/JarneRenders/K2-Hamiltonian-Graphs.

[57] J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. Network fault costs based on minimum leaf spanning trees. *arXiv preprint*, 2025. arXiv: 2502.10213.

[58] J. Goedgebeur, J. Renders and C. T. Zamfirescu. GenK2Hypohamiltonian [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/GenK2Hypohamiltonian.

[59] J. Goedgebeur and C. T. Zamfirescu. Improved bounds for hypohamiltonian graphs. *Ars Math. Contemp.*, 13(2):235–257, 2017. DOI: 10.26493/1855-3974.1044.eaa.

[60] J. Goedgebeur and C. T. Zamfirescu. On hypohamiltonian snarks and a theorem of Fiorini. *Ars Math. Contemp.*, 14(2):227–249, 2018. DOI: 10.26493/1855-3974.1176.9b7.

[61] J. Goedgebeur, J. Jooken, O.-H. S. Lo, B. Seamone and C. T. Zamfirescu. Few hamiltonian cycles in graphs with one or two vertex degrees. *Math. Comput.*, 93(350):3059–3082, 2024. DOI: 10.1090/mcom/3943.

[62] J. Goedgebeur, E. Máčajová and J. Renders. The Frank number and nowhere-zero flows on graphs. *Eur. J. Comb.*, 126:104127, 2025. DOI: 10.1016/j.ejc.2025.104127.

[63] J. Goedgebeur, A. Neyt and C. T. Zamfirescu. Structural and computational results on platypus graphs. *Appl. Math. Comput.*, 386:125491, 2020. DOI: 10.1016/j.amc.2020.125491.

[64] J. Goedgebeur and J. Renders. Generation of cycle permutation graphs and permutation snarks. In R. Královič and V. Kůrková, editors, *SOFSEM 2025: Theory and Practice of Computer Science* (Bratislava, Slovakia), volume 15538 of *Lecture Notes in Computer Science*, pages 333–346. Springer, Cham, 2025. DOI: 10.1007/978-3-031-82670-2_24.

[65] J. Goedgebeur, J. Renders and S. Van Overberghe. Generation of cycle permutation graphs and permutation snarks. *arXiv preprint*, 2025. arXiv: 2411.12606.

[66] J. Goedgebeur, J. Renders and C. T. Zamfirescu. Generation and new infinite families of K_2-hypohamiltonian graphs. *Discret. Math.*, 347(7):113981, 2024. DOI: 10.1016/j.disc.2024.113981.

[67] R. J. Gould. Recent advances on the hamiltonian problem: survey III. *Graphs Comb.*, 30:1–46, 2014. DOI: 10.1007/s00373-013-1377-x.

[68] R. L. Graham, M. Grötschel and L. Lovász. *Handbook of Combinatorics*, volume 1. Elsevier, 1995.

[69] E. J. Grinberg. Plane homogeneous graphs of degree three without hamiltonian circuits. *Latv. Math. Yearb., Izdat. Zinatne, Riga*, 4:51–58, 1968. (In Russian) English Translation by D. Zeps: https://arxiv.org/abs/0908.2563.

[70] M. Grohe and P. Schweitzer. The graph isomorphism problem. *Commun. ACM*, 63(11):128–134, 2020. DOI: 10.1145/3372123.

[71] M. Grötschel. On the monotone symmetric travelling salesman problem: hypohamiltonian/hypotraceable graphs and facets. *Math. Oper. Res.*, 5(2):161–320, 1980. DOI: 10.1287/moor.5.2.285.

[72] M. Grötschel and Y. Wakabayashi. On the structure of the monotone asymmetric travelling salesman polytope I: hypohamiltonian facets. *Discret. Math.*, 34(1):43–59, 1981. DOI: 10.1016/0012-365X(81)90021-2.

[73] B. Grünbaum. Vertices missed by longest paths or circuits. *J. Comb. Theory, Ser. A*, 17(1):31–38, 1974. DOI: 10.1016/0097-3165(74)90025-9.

[74] F. Harary and G. Prins. The number of homeomorphically irreducible trees, and other species. *Acta Math.*, 101(1-2):141–162, 1959. DOI: 10.1007/BF02559543.

[75] J. C. Herz, J. J. Duby and F. Vigué. Recherche systématique des graphes hypohamiltoniens. In P. Rosenthiehl, editor, *Theory of Graphs: International Symposium, Rome, July 1966*, pages 153–159, Paris, France. Dunod, 1967. In French.

[76] A. Hoffmann-Ostenhof, K. Noguchi and K. Ozeki. On homeomorphically irreducible spanning trees in cubic graphs. *J. Graph Theory*, 89(2):93–100, 2018. DOI: 10.1002/jgt.22242.

[77] D. A. Holton. Two open problems in graph theory. *New Zealand J. Math.*, 22(1):67–78, 1993.

[78] D. A. Holton and J. Sheehan. *The Petersen graph*. In volume 7. Australian Mathematical Society Lecture Series. Cambridge University Press, New York, NY, USA, 1993. Chapter 7: Hypohamiltonian graphs. DOI: 10.1017/CBO9780511662058.

[79] F. Hörsch and Z. Szigeti. Connectivity of orientations of 3-edge-connected graphs. *Eur. J. Comb.*, 94:103292, 2021. DOI: 10.1016/j.ejc.2020.103292.

[80] L.-H. Hsu and C.-K. Lin. *Graph Theory and Interconnection Networks*. CRC Press, Boca Raton, FL, USA, 1st edition, 2008. DOI: 10.1201/9781420044829.

[81] R. Isaacs. Infinite families of nontrivial trivalent graphs which are not Tait colorable. *Am. Math. Mon.*, 82(3):221–239, 1975. DOI: 10.2307/2319844.

[82] F. Jaeger. A Survey of the Cycle Double Cover Conjecture. In B. R. Alspach and C. D. Godsil, editors, *North-Holland Mathematics Studies*. Volume 115, Annals of Discrete Mathematics (27): Cycles in Graphs, pages 1–12. North-Holland, 1985. DOI: 10.1016/S0304-0208(08)72993-1.

[83] F. Jaeger. On vertex-induced forests in cubic graphs. In *Proceedings 5th Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume 10 of *Congr. Numer.* Pages 501–512. Utilitas Mathematics Pub., 1974.

[84] P. Joffe. *Some properties of 3-polytopal graphs*. PhD thesis, City University of New York, New York, NY, USA, 1982.

[85] M. Jooyandeh, B. D. McKay, P. R. J. Östergård, V. H. Pettersson and C. T. Zamfirescu. Planar hypohamiltonian graphs on 40 vertices. *J. Graph Theory*, 84(2):121–133, 2017. DOI: 10.1002/jgt.22015.

[86] S. Kapoor and H. Ramesh. Algorithms for enumerating all spanning trees of undirected and weighted graphs. *SIAM J. Comput.*, 24(2):247–265, 1995. DOI: 10.1137/S009753979225030X.

[87] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher and J. D. Bohlinger, editors, *Complexity of Computer Computations*, IBM research symposia series, pages 85–103. Springer US, Boston, MA, USA, 1972. DOI: 10.1007/978-1-4684-2001-2_9.

[88] G. O. H. Katona, A. Kostochka, J. Pach and B. S. Stechkin. Locally hamiltonian graphs. *Math. Notes Acad. Sci. USSR*, 45(1):25–29, 1989. DOI: 10.1007/BF01158712. (36–42 in the original Math. Zametki **45**(1).)

[89] V. Klee. Which generalized prisms admit H-circuits? In Y. Alavi, D. R. Lick and A. T. White, editors, *Graph Theory and Applications*, volume 303 of *Lecture Notes in Mathematics*, pages 173–178, Berlin, Germany. Springer, 1972. DOI: 10.1007/BFb0067368.

[90] D. Kőnig. Gráfok és alkalmazásuk a determinánsok Žs a halmazok elméletére [Graphs and their applications to the theory of determinants and sets]. *Mat. és Termtud. Ért.*, 34:104–119, 1916. In Hungarian.

[91] D. Kratsch, J. Lehel and H. Müller. Toughness, hamiltonicity and split graphs. *Discret. Math.*, 150(1–3):231–245, 1996. DOI: 10.1016/0012-365X(95)00190-8.

[92] H.-S. Lim, H.-C. Kim and J.-H. Park. Hypohamiltonian-connectedness and pancyclicity of hypercube-like interconnection networks with faulty elements. In *Proceedings of the Workshop on Algorithms and Computation WAAC2005*, Seoul, Korea, 2005.

[93] W. F. Lindgren. An infinite class of hypohamiltonian graphs. *Am. Math. Mon.*, 74(9):1087–1089, 1967. DOI: 10.2307/2313617.

[94] G. Liva, B. Matuz, E. Paolini and M. Chiani. Short non-binary IRA codes on large-girth hamiltonian graphs. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 2606–2610, Ottawa, Canada, 2012. DOI: 10.1109/ICC.2012.6363976.

[95] H.-I. Lu and R. Ravi. Approximation for maximum leaf spanning trees in almost linear time. *J. Algorithms*, 29(1):132–141, 1998. DOI: 10.1006/jagm.1998.0944.

[96] E. Máčajová and M. Škoviera. Permutation snarks of order 2 (mod 8). *Acta Math. Univ. Comen.*, 88(3):929–934, 2019.

[97] J. Malkevitch. Spanning trees in polytopal graphs. *Ann. New York Acad. Sci.*, 319:362–367, 1979. DOI: 10.1111/j.1749-6632.1979.tb32810.x.

[98] B. D. McKay. Hypohamiltonian cubic planar graphs with girth 5. *J. Graph Theory*, 85(1):7–11, 2017. DOI: 10.1002/jgt.22043.

[99] B. D. McKay. Isomorph-free exhaustive generation. *J. Algorithms*, 26(2):306–324, 1998. DOI: 10.1006/jagm.1997.0898.

[100] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. DOI: 10.1016/j.jsc.2013.09.003.

[101] C. S. J. A. Nash-Williams. On orientations, connectivity and odd-vertex-pairings in finite graphs. *Can. J. Math.*, 12:555–567, 1960. DOI: 10.4153/CJM-1960-049-6.

[102] D. A. Nelson. *Hamiltonian graphs*. Master's thesis, Vanderbilt University, Nashville, TN, USA, 1973.

[103] R. Nomura and S. Tsuchiya. Plane graphs without homeomorphically irreducible spanning trees. *Ars Comb.*, 141:157–165, 2018.

[104] O. Ore. Note on hamilton circuits. *Am. Math. Mon.*, 67(1):55–55, 1960. DOI: 10.2307/2308928.

[105] K. Ozeki. Personal communication, May 2019.

[106] K. Ozeki, G. Wiener and C. T. Zamfirescu. On minimum leaf spanning trees and a criticality notion. *Discret. Math.*, 343(7):111884, 2020. DOI: 10.1016/j.disc.2020.111884.

[107] J. Pach and G. Tóth. Which crossing number is it anyway? *J. Comb. Theory, Ser. B.*, 80(2):225–246, 2000. DOI: 10.1006/jctb.2000.1978.

[108] F. Pfender. Hamiltonicity and forbidden subgraphs in 4-connected graphs. *J. Graph Theory*, 49(4):262–272, 2005. DOI: 10.1002/jgt.20080.

[109] T. Pisanski and J. Shawe-Taylor. Cycle permutation graphs with large girth. *Glas. Mat.*, 17(2):233–236, 1982.

[110] T. Pisanski and J. Shawe-Taylor. Search for minimal trivalent cycle permutation graphs with girth nine. *Discret. Math.*, 36(1):113–115, 1981. DOI: 10.1016/0012-365X(81)90179-5.

[111] S. Radziszowski. Small Ramsey numbers. *Electron. J. Comb.* DOI: 10.37236/21. DS1: Sep 6–2024.

[112] R. C. Read. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. In B. Alspach, P. Hell and D. J. Miller, editors, *Annals of Discrete Mathematics*. Volume 2, Algorithmic Aspects of Combinatorics, pages 107–120. Elsevier, 1978. DOI: 10.1016/S0167-5060(08)70325-X.

[113] N. Robertson, D. Sanders, P. Seymour and R. Thomas. The four-colour theorem. *J. Comb. Theory, Ser. B*, 70(1):2–44, 1997. DOI: 10.1006/jctb.1997.1750.

[114] V. Rosta. Ramsey theory applications. *Electron. J. Comb.* DOI: 10.37236/34. DS13: Dec 7–2004.

[115] M. Z. Rozenfel'd. The construction and properties of certain classes of strongly regular graphs. *Uspehi Mat. Nauk*, 28(3):197–198, 1973. In Russian.

[116] G. Salamon and G. Wiener. On finding spanning trees with few leaves. *Inform. Proc. Lett.*, 105(5):164–169, 2008. DOI: 10.1016/j.ipl.2007.08.030.

[117] D. P. Sanders. On hamilton cycles in certain planar graphs. *J. Graph Theory*, 21(1):43–50, 1996. DOI: 10.1002/(SICI)1097-0118(199601)21:1<43::AID-JGT6>3.0.CO;2-M.

[118] U. Schmidt. *Überprüfung des Beweises für den Vierfarbensatz*. Master's thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, Germany, 1982. In German.

[119] P. D. Seymour. Nowhere-zero 6-flows. *J. Comb. Theory, Ser. B*, 30(2):130–135, 1981. DOI: 10.1016/0095-8956(81)90058-7.

[120] P. D. Seymour. On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. *Proceedings of the London Mathematical Society*, s3-38(3):423–460, 1979. DOI: 10.1112/plms/s3-38.3.423.

[121] P. D. Seymour. Sums of circuits. In J. A. Bondy and U. R. S. Murty, editors, *Graph Theory and Related Topics*, pages 341–355, New York, NY, USA. Academic Press, 1979.

[122] R. Solis-Oba. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In G. Bilardi, G. F. Italiano, A. Pietracaprina and G. Pucci, editors, *Proceedings of the 6th ESA Symposium*, volume 1461 of *Lecture Notes in Computer Science*, pages 441–452, Berlin, German. Springer, 1998. DOI: 10.1007/3-540-68530-8_37.

[123] R. Sousselier. *Problème no. 29: Le cercle des irascibles*. In *Problèmes plaisants et délectables*. Volume 7. Rev. Franç. Rech. Opérationnelle. In French. 1963, pages 405–406.

[124] R. Staden. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Res.*, 6(7):2601–2610, 1979. DOI: 10.1093/nar/6.7.2601.

[125] G. Szekeres. Polyhedral decompositions of cubic graphs. *Bull. Aust. Math. Soc.*, 8(3):367–387, 1973. DOI: 10.1017/S0004972700042660.

[126] R. Thomas and X. Yu. 4-connected projective-planar graphs are hamiltonian. *J. Comb. Theory, Ser. B*, 62(1):114–132, 1994. DOI: 10.1006/jctb.1994.1058.

[127] A. G. Thomason. Hamiltonian cycles and uniquely edge colourable graphs. *Ann. Discret. Math.*, 3:259–268, 1978. DOI: 10.1016/S0167-5060(08)70511-9.

[128] C. Thomassen. Hypohamiltonian graphs and digraphs. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 557–571, Berlin, Germany. Springer, 1978.

[129] C. Thomassen. On hypohamiltonian graphs. *Discret. Math.*, 10(2):383–390, 1974. DOI: 10.1016/0012-365X(74)90128-9.

[130] C. Thomassen. Planar and infinite hypohamiltonian and hypotraceable graphs. *Discret. Math.*, 14(4):377–389, 1976. DOI: 10.1016/0012-365X(76)90071-6.

[131] C. Thomassen. Planar cubic hypohamiltonian and hypotraceable graphs. *J. Comb. Theory, Ser. B*, 30(1):36–44, 1981. DOI: 10.1016/0095-8956(81)90089-7.

[132] W. T. Tutte. A contribution to the theory of chromatic polynomials. *Can. J. Math.*, 6:80–91, 1954. DOI: 10.4153/CJM-1954-010-9.

[133] W. T. Tutte. A theorem on planar graphs. *Trans. Am. Math. Soc.*, 82(1):99–116, 1956. DOI: 10.2307/1992980.

[134] N. Van Cleemput and C. T. Zamfirescu. Regular non-hamiltonian polyhedral graphs. *Appl. Math. Comput.*, 338:192–206, 2018. DOI: 10.1016/j.amc.2018.05.062.

[135] F. Van de Steene. *Het minimum bladgetal van grafen.* Master's thesis, Ghent University, Ghent, Belgium, 2020. J. Goedgebeur and C. T. Zamfirescu (supervisors). In Dutch.

[136] S. A. van Aardt, A. P. Burger, M. Frick, B. Llano and R. Zuazua. Infinite families of 2-hypohamiltonian/2-hypotraceable oriented graphs. *Graphs Comb.*, 30:783–800, 2014. DOI: 10.1007/s00373-013-1312-1.

[137] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.*, 3:25–30, 1964.

[138] J.-J. Wang, C.-N. Hung and L.-H. Hsu. Optimal 1-hamiltonian graphs. *Inform. Process. Lett.*, 65(3):157–161, 1998. DOI: 10.1016/S0020-0190(98)00004-0.

[139] H. Whitney. 2-Isomorphic graphs. *Am. J. Math.*, 55(1):245–254, 1933. DOI: 10.2307/2371127.

[140] G. Wiener. Leaf-critical and leaf-stable graphs. *J. Graph Theory*, 84(4):443–459, 2017. DOI: 10.1002/jgt.22034.

[141] G. Wiener. New constructions of hypohamiltonian and hypotraceable graphs. *J. Graph Theory*, 87(4):526–535, 2018. DOI: 10.1002/jgt.22173.

[142] G. Wiener and M. Araya. On planar hypohamiltonian graphs. *J. Graph Theory*, 67(1):55–68, 2011. DOI: 10.1002/jgt.20513.

[143] C. T. Zamfirescu. K_2-hamiltonian graphs: I. *SIAM J. Discret. Math.*, 35(3):1706–1728, 2021. DOI: 10.1137/20M1355252.

[144] C. T. Zamfirescu. On hypohamiltonian and almost hypohamiltonian graphs. *J. Graph Theory*, 79(1):63–81, 2015. DOI: 10.1002/jgt.21815.

[145] C. T. Zamfirescu. On non-hamiltonian graphs for which every vertex-deleted subgraph is traceable. *J. Graph Theory*, 86(2):223–243, 2017. DOI: 10.1002/jgt.22122.

[146] C. T. Zamfirescu. On platypus graphs and the Steiner-Deogun property. Submitted for publication.

[147] C. T. Zamfirescu and T. I. Zamfirescu. Every graph occurs as an induced subgraph of some hypohamiltonian graph. *J. Graph Theory*, 88(4):551–557, 2018. DOI: 10.1002/jgt.22228.

[148] C.-Q. Zhang. *Integer flows and cycle covers of graphs*, number 205 in Monographs and textbooks in pure and applied mathematics. Marcel Dekker, New York, 1997.

Statement on the use of Generative AI

I did not use generative AI assistance tools during the research/writing process of my thesis.
The text, code, and images in this thesis are my own (unless otherwise specified). Generative AI has only been used in accordance with the KU Leuven guidelines and appropriate references have been added. I have reviewed and edited the content as needed and I take full responsibility for the content of the thesis.

List of Publications

Journal articles

J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. K_2-hamiltonian graphs: II. *J. Graph Theory*, 105(4):580–611, 2024. DOI: 10.1002/jgt.23057

J. Goedgebeur, J. Renders and C. T. Zamfirescu. Generation and new infinite families of K_2-hypohamiltonian graphs. *Discret. Math.*, 347(7):113981, 2024. DOI: 10.1016/j.disc.2024.113981

J. Renders, S.-I. Tokunaga and C. T. Zamfirescu. Domination of triangulated discs and maximal outerplanar graphs. *Appl. Math. Comput.*, 473:128648, 2024. DOI: 10.1016/j.amc.2024.128648

J. Goedgebeur, E. Máčajová and J. Renders. The Frank number and nowhere-zero flows on graphs. *Eur. J. Comb.*, 126:104127, 2025. DOI: 10.1016/j.ejc.2025.104127

E. Carlson, W. Fletcher, M. Montee, C. Nguyen, J. Renders and X. Zhang. Graphs with many hamiltonian paths. *To appear in Involve*, 2025+

Conference publications

J. Goedgebeur, E. Máčajová and J. Renders. On the frank number and nowhere-zero flows on graphs. In D. Paulusma and B. Ries, editors, *Proceedings of the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG2023)* (Fribourg, Switzerland), volume 14093 of *Lecture Notes in Computer Science*, pages 363–375. Springer, Cham., 2023. DOI: 10.1007/978-3-031-43380-1_26

J. Goedgebeur and J. Renders. Generation of cycle permutation graphs and permutation snarks. In R. Královič and V. Kůrková, editors, *SOFSEM 2025: Theory and Practice of Computer Science* (Bratislava, Slovakia), volume 15538 of *Lecture Notes in Computer Science*, pages 333–346. Springer, Cham, 2025. DOI: 10.1007/978-3-031-82670-2_24

Preprints (submitted for publication)

J. Goedgebeur, K. Noguchi, J. Renders and C. T. Zamfirescu. HIST-critical graphs and Malkevitch's conjecture. *arXiv preprint*, 2024. arXiv: 2401.04554

J. Goedgebeur, D. Mattiolo, G. Mazzuoccolo, J. Renders and I. H. Wolf. Cubic graphs with edges in exactly one perfect matching. *arXiv preprint*, 2024. arXiv: 2402.08538

J. Goedgebeur, D. Mattiolo, G. Mazzuoccolo, J. Renders, L. Toffanetti and I. H. Wolf. On the existence of factors intersecting sets of cycles in regular graphs. *arXiv preprint*, 2024. arXiv: 2411.09806

M. Daneels, J. Goedgebeur and J. Renders. Infinite families of planar graphs of a given injective chromatic number. *arXiv preprint*, 2024. arXiv: 2412.09969

J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. Network fault costs based on minimum leaf spanning trees. *arXiv preprint*, 2025. arXiv: 2502.10213

J. Goedgebeur, J. Renders and S. Van Overberghe. Generation of cycle permutation graphs and permutation snarks. *arXiv preprint*, 2025. arXiv: 2411.12606

Open Source Software

J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. K2-Hamiltonian Graphs [Computer software], version 1, 2022. URL: https://github.com/JarneRenders/K2-Hamiltonian-Graphs

J. Goedgebeur, J. Renders and C. T. Zamfirescu. GenK2Hypohamiltonian [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/GenK2Hypohamiltonian

J. Goedgebeur, E. Máčajová and J. Renders. Frank-Number [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/Frank-Number

J. Goedgebeur, K. Noguchi, J. Renders and C. T. Zamfirescu. HistChecker [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/HistChecker

J. Renders, S.-I. Tokunaga and C. T. Zamfirescu. Domination [Computer software], version 1, 2023. URL: https://github.com/JarneRenders/domination

J. Goedgebeur, D. Mattiolo, G. Mazzuoccolo, J. Renders and I. H. Wolf. Lonely Edges (version 1) [Computer software], 2023. URL: https://github.com/JarneRenders/lonelyEdges

J. Renders. CircumferenceChecker (version 1) [Computer software], 2024. URL: https://github.com/JarneRenders/CircumferenceChecker

J. Goedgebeur, J. Renders and S. Van Overberghe. Cycle Permutation Graphs [Computer software], version 2, 2025. URL: https://github.com/JarneRenders/Cycle-Permutation-Graphs

J. Goedgebeur, J. Renders, G. Wiener and C. T. Zamfirescu. FaultCost [Computer software], version 1, 2025. URL: https://github.com/JarneRenders/faultCost

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
ALGORITHMIC GRAPH THEORY
Etienne Sabbelaan 53
B-8500 Kortrijk
https://kulak.kuleuven.be/algo