Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Hamiltonicity of cages

Bachelor Thesis

2025

Jakub Šmihuľa

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Hamiltonicity of cages

Bachelor Thesis

Study Programme:	Applied Computer Science
Field of Study:	Computer Science
Department:	Department of applied informatics
Supervisor:	doc. RNDr. Tatiana Jajcayová, PhD.

Bratislava, 2025
Jakub Šmihuľa

95996957

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Jakub Šmihuľa
Študijný program:	aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Hamiltonicity of Cages
Hamiltonovskosť klietok

Anotácia: (k,g)-graf je regulárny graf valencie k a dĺžky najkratšieho cyklu g. Je známe, že pre každú dvojicu parametrov (k,g) existuje nekonečne veľa grafov. Graf najmenšieho rádu n(k,g) medzi týmito grafmi sa nazýva (k,g)-klietka. Nájsť (k,g)-klietku pre dané parametre je takzvaný Cage Problem. Klietky sú známe len pre niekoľko párov parametrov. Je výpočtovo veľmi ťažké dokázať, že nejaký (k,g)-graf je klietka. Najmenší známy (k,g)-graf, o ktorom ešte nebolo dokázané, že je klietka, sa nazýva rekordný rec(k,g)-graf. Konštrukcie klietok a rekordných grafov, štúdium ich vlastností a variácie problému klietok je veľmi aktívna oblasť výskumu.

Cieľ: Sachs vo svojej práci predpokladal, že pre každú dvojicu parametrov k,g existuje klietka, ktorá je hamiltonovská. Cieľom bakalárske práce je navrhnúť, zbehnúť a analyzovať experimenty na známych klietkach a rekordných grafoch a analyzovať ich hamiltonovskosť.

Literatúra:
G. Exoo and R. Jajcay. Dynamic cage survey. The Electronic Journal of Combinatorics, pages
DS16–Jul, 2012.
T. B. Jajcayov´a, S. Filipovski, and R. Jajcay. Counting cycles in graphs with small excess. Lecture Notes of Seminario Interdisciplinare di Matematica, 2016.
P. Erd˝os and H. Sachs. Regul¨are graphen gegebener taillenweite mit minimaler knotenzahl.Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe, 12(251-257):22, 1963.

Vedúci:	doc. RNDr. Tatiana Jajcayová, PhD.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	doc. RNDr. Tatiana Jajcayová, PhD.
Dátum zadania:	24.09.2024

Dátum schválenia: 11.11.2024

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.... študent

.... vedúci práce

95996957

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Jakub Šmihuľa
Study programme:	Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Hamiltonicity of Cages

Annotation: A (k,g)-graph is a regular graph of valency k and length of shortest cycle g. It was shown that for each pair of parameters (k,g) there exists infinitely many graphs. The graph of the smallest order, which we will denote n(k,g), among these graphs is called a (k,g)-cage. To find a (k,g)-cage for given parameters is so called Cage Problem. Cages are known only for few pairs of parameters. It is computationally very difficult to prove that some (k,g)-graph is a cage. The smallest known (k,g)-graph, that was not yet proven to be a cage is called a record rec(k,g)-graph. Constructions of cages and record graphs, studying their properties and variations to Cage problem is very active research area.

Aim: Sachs in his work conjectured that for each pair of parameters k,g, there exists a cage that is Hamiltonian. The aim of the theses is to design, run and analyze experiments on known cages and record graphs and analyze their Hamiltonicity.

Literature: G. Exoo and R. Jajcay. Dynamic cage survey. The Electronic Journal of Combinatorics, pages
DS16–Jul, 2012.
T. B. Jajcayov´a, S. Filipovski, and R. Jajcay. Counting cycles in graphs with small excess.Lecture Notes of Seminario Interdisciplinare di Matematica, 2016.
P. Erd˝os and H. Sachs. Regul¨are graphen gegebener taillenweite mit minimaler knotenzahl.Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe, 12(251-257):22, 1963.

Supervisor:	doc. RNDr. Tatiana Jajcayová, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	24.09.2024

Approved: 11.11.2024 doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

.... Student

.... Supervisor

Acknowledgments: I would like to thank my supervisor, RNDr. Tatiana Jajcayová, PhD., for her helpful advice and support during my work on this bachelor thesis.

Part of the research was obtained using the computational resources procured in the national project *National competence centre for high performance computing* within the Operational programme Integrated infrastructure (project code: p1310-25-t).

Abstrakt

Cieľom tejto práce je skúmať hamiltonovskosť v klietkach a rekordných grafoch. Najprv sme overili známe postačujúce podmienky pre existenciu hamiltonovského cyklu vo všeobecných, ako aj v regulárnych grafoch. Keďže tieto podmienky neboli splnené, využili sme softvér na hľadanie hamiltonovských kružníc, ktoré sme následne analyzovali a vizualizovali. Ďalej sme implementovali známu konštrukciu, ktorá generuje hamiltonovské k-regulárne grafy s daným obvodom, a porovnali sme ich veľkosť so známymi klietkami. Na záver sme sa priklonili k hypotéze, že všetky klietky sú Hamiltonovské, s výnimkou známeho prípadu Petersenovho grafu.

Kľúčové slová: graf, hamiltonovská kružnica, klietka, rekordný graf

Abstract

The aim of this thesis is to study Hamiltonicity in cages and record graphs. First, we verified known sufficient conditions for the existence of a Hamiltonian cycle in both general and regular graphs. Since these conditions were not satisfied, we used software to search for Hamiltonian cycles, which we then analyzed and visualized. Furthermore, we implemented a known construction that generates Hamiltonian k-regular graphs with a given girth, and compared their sizes with known cages. In conclusion, we supported the hypothesis that all cages are Hamiltonian, except for the well-known case of the Petersen graph.

Keywords: graph, Hamiltonian cycle, cage, record graph

Contents

List of Figures

List of Tables

Introduction

Studying the Hamiltonicity of a graph is a very interesting problem, as it is non-trivial and has many real-life applications in areas such as networks and topology, logistics, and electronic systems. Our thesis focuses on finding Hamiltonian cycles in cages. Although graphs like cages are well studied, their Hamiltonicity is still not proven. Earlier work on this topic was done by Geoffrey Exoo, but his experiments were never published. This problem is particularly interesting because cages appear in many graph constructions due to their minimality and regularity, and their Hamiltonicity could be useful in building these constructions. An example of such construction can be found in mixed graphs.

Furthermore, by testing Hamiltonicity on cages and then on record graphs, we may begin to observe certain patterns. These observations could support a hypothesis that all cages are Hamiltonian. Comparing both types of graphs side by side allows us to explore this idea more deeply and could lead to the conclusion that while cages are Hamiltonian, some record graphs may not be. This might suggest the existence of smaller Hamiltonian record graphs, which would be valuable for further study.

In the first chapter, we introduce the basic terms and concepts from graph theory that are essential for understanding cages and Hamiltonian cycles. We also review common ways to represent graphs, since these representations will be used throughout the rest of the thesis.

The second chapter is dedicated to Hamiltonian cycles. We present several sufficient conditions under which a graph is guaranteed to be Hamiltonian and explore heuristic methods used in finding Hamiltonian cycles. It will also explain the differences between complexity classes of problems.

The third chapter will define cages as well as record graphs. It will also present some known cages and known record graphs. Additionally, it will describe an algorithm for constructing Hamiltonian cages and compare these constructions to known cages.

In the fourth chapter we present the software and hardware tools used for finding Hamiltonian cycles in graphs, as well as their performance.

Finally, in the fifth chapter, we present our experimental results. We explain the origin of the datasets we used and how we transformed them into a consistent format. We describe how we reduced the Hamiltonian cycle problem to the Travelling Salesman

Problem, and the structure of the output results. Then, we analyze how sufficient conditions for Hamiltonicity apply to both cages and record graphs. The chapter concludes with our results, including any limitations and how the experimental data is stored.

Chapter 1

Basic Definitions

In this chapter, we will present some basic definitions and terminology of graph theory that will be used in this thesis.

1.1 Basic of Graph theory

This section contains definitions of graph theory that are key to understanding the concept of Hamiltonicity and cages.

1.1.1 Graph

Graph is a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$. E is 2 element subset of V. Elements of set V are called vertices of graph G, and elements of E are called edges of graph G.

A vertex v is incident with an edge e if $v \in e$. They are also called neighbors. Two vertices v, u are adjacent if they are incident to the same edge.

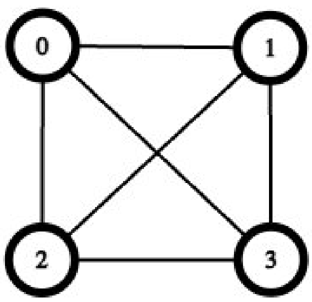


Figure 1.1: Connected undirected simple graph

In this work, we are working with connected undirected simple graphs. Therefore, edges do not have a direction, two vertices are connected by just one edge, and no edge starts and ends in the same vertex (such an edge is called a self-loop) [29] In contrast to a simple graph, a multigraph allows multiple edges between the same pair of vertices.

The vertex degree of vertex v is denoted as $deg(v)$, which is the number of edges incident with v. Since the graph is simple, it is also the number of neighbors of v.

A regular graph is a graph where all vertices have the same degree. A k-regular graph is a graph where all vertices have a degree equal to k.[29]

A symmetric graph is one in which, for any two pairs of adjacent vertices, there exists an automorphism (a symmetry of the graph) that maps one pair to the other.[13]

The walk in G is a finite non-null sequence $W = v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k$ whose terms are alternately vertices and edges, such that, for $i \leq 1 \leq k$, the ends of e_i are v_{i-1} and v_i. The integer k is the length of W [5]

If the edges $e_1, e_2, ..., e_k$ of a walk W are distinct, W is called a trial.

If the vertices $v_0, v_1, ...v_k$ of a walk W are distinct, W is called a path.

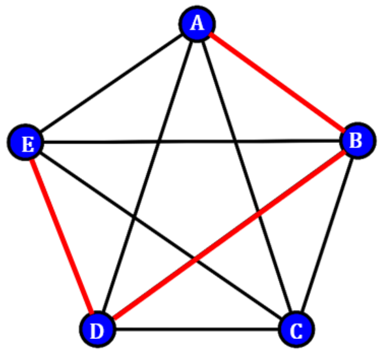


Figure 1.2: K_5 graph

On 1.2 is highlighted path from vertex A to vertex E. As seen in the figure 1.2, the path could be shorter.

The Shortest path is the path between two vertices in a simple graph that crosses the least number of edges.

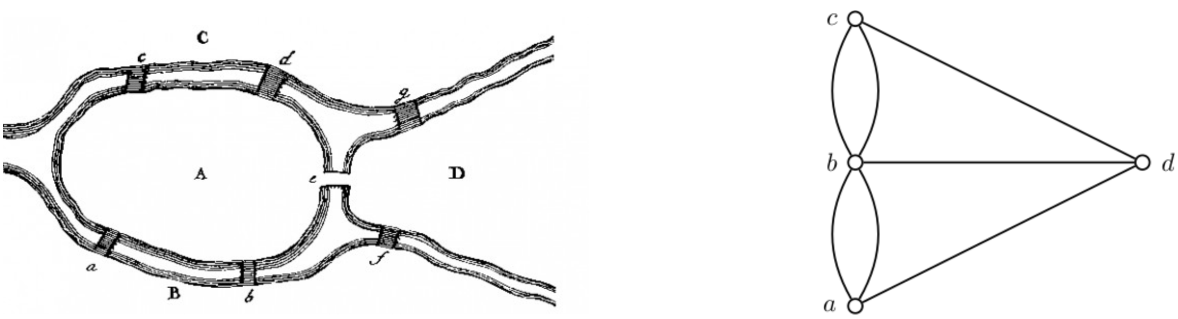


Figure 1.3: The Seven Bridges of Königsberg and corresponding graph

The circuit is a non-empty trial in which the first and last vertices are equal.

The cycle is a non-empty path in which the first and last vertices are equal.

Length of a cycle is defined as the number of edges it contains, which is equal to the number of vertices in the cycle.

A graph is connected if there is a path between any pair of vertices.

The tour of G is a closed walk that traverses each edge of G at least once.

The Euler tour is a tour that traverses each edge exactly once.

A path that contains every vertex of graph G is called **Hamilton path** of G, and a cycle that contains every vertex of graph G is called a **Hamilton cycle**.

With topic of Hamiltonian cycles is related Travelling salesman problem (TSP). TSP asks for finding the shortest possible route that visits each vertex exactly once and returns to the starting vertex.

1.2 Graph Metrics

The eccentricity of a vertex v is the maximum distance between v and any other vertex u in the graph. The distance between two vertices is the length of the shortest path that connects v and u.

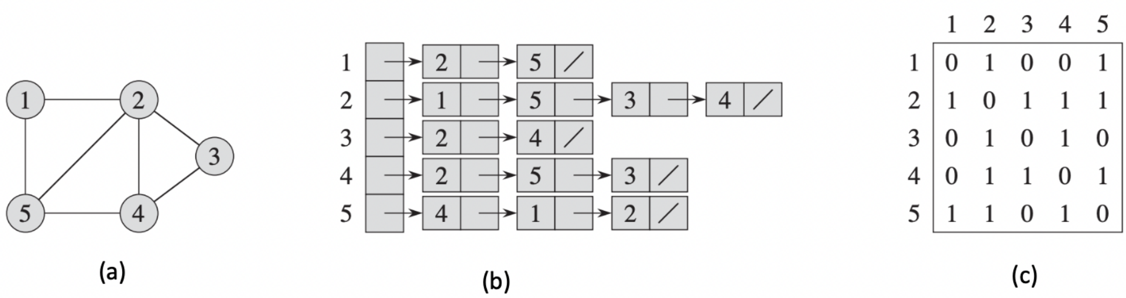
The diameter of the graph is defined as the maximum eccentricity of a vertex that can be found in a graph.

On the other hand, the radius of a graph is the minimum eccentricity of a vertex in a graph.

Girth is the length of the shortest cycle contained in a graph.

1.3 Graph representations

Our work also consists of the implementation section therefore, it is necessary to mention representations of graphs.



$$G = \{(1,2);(1,5);(2,5);(2,4);(2,3);(3,4);(3,5)\}$$

(d)

Figure 1.4: Representations of a graph[7]

1.3.1 Set of Edges

This graph representation is among the most fundamental, as graph G is defined as a set of vertex pairs.

However, this representation is not particularly efficient. The process of determining whether an edge exists between two given vertices requires a lookup operation with a time complexity of $O(|E|)$, where $|E|$ denotes the number of edges in graph G. Conversely, the memory space required for this representation is $\Theta(|E|)$, which is relatively efficient compared to other representations.

An example of this representation is illustrated in Figure 1.4 (d), where the graph from Figure 1.4 (a) is represented as a set of edges.

1.3.2 Adjacency-List Representation

This graph representation consists of an array of adjacency lists, where each vertex maintains a list of its adjacent vertices. If an edge exists between two vertices, say (v, u), then $u \in Adj[v]$ and similarly, $v \in Adj[u]$ in the case of an undirected graph. [7]

This representation offers improved performance in terms of time complexity for checking the existence of an edge between two vertices. Specifically, the time complexity of this operation is $O(\deg(v))$, where $\deg(v)$ denotes the degree of vertex v in graph G. The amount of memory which is required for this is $\Theta(|E| + |V|)$.

An example of this representation is illustrated in Figure 1.4 (b), where the graph from Figure 1.4 (a) is represented as an adjacency list.

1.3.3 Adjacency-matrix representation

An adjacency matrix is a $|V| \times |V|$ matrix, where $|V|$ represents the number of vertices in the graph. This representation follows a simple principle: if an edge exists between two vertices, it is denoted by 1, whereas the absence of an edge is represented by 0.

One of the key advantages of this representation is its efficiency in edge lookup operations, achieving a time complexity of $\Theta(1)$, as checking the existence of an edge requires only accessing the corresponding matrix entry. However, the space complexity is not optimal, as it requires $\Theta(|V|^2)$ memory, which can become problematic for large graphs.

An example of this representation is illustrated in Figure 1.4 (c), where the graph from Figure 1.4 (a) is represented as an adjacency matrix.

Chapter 2

Hamilton Cycles

2.1 History

This term originated from a game called Icosian, which was invented by Sir William Rowan Hamilton in 1859.[12] The main task of the puzzle is to find a cycle on a wooden dodecahedron with 20 vertices in such a way that every vertex is visited only once. Since then, this problem has been formulated and later studied all around the world. Now, it is known as an NP-complete problem, and therefore, finding a solution is very desirable.

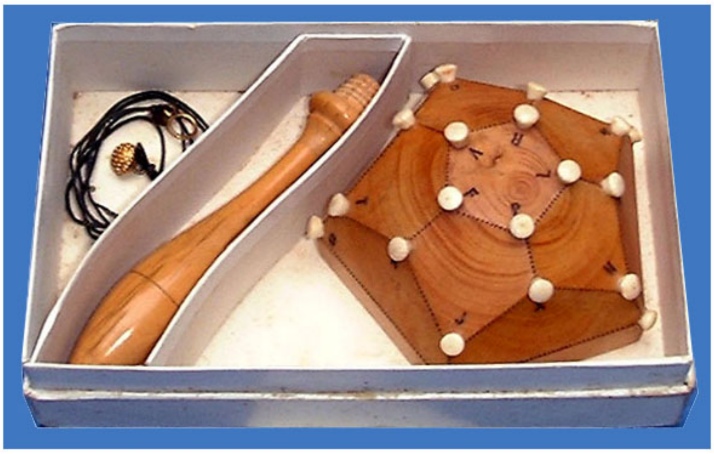


Figure 2.1: Icosian game

2.2 Definition

If $G = (V, E)$ is a graph with $|V| \geq 3$, then G has a Hamilton cycle if there exists a cycle in G that contains every vertex in V. Hamiltonian path is similarly a path in G that contains each vertex. [14]

2.2.1 General sufficient conditions

We present two well-known sufficient conditions for a graph to have a Hamilton cycle. The sufficient condition is when a graph satisfies the condition, then it is Hamiltonian, but there exist graphs that do not meet a condition but still are Hamiltonian.

Lemma 1. *Every graph with $|V| = n \geq 3$ and minimum degree at least $n/2$ is connected.*

Proof. Assume by contradiction, that graph G is not connected. Therefore it has two components C_1 and C_2. Since each vertex $v \in C_1$ has degree at least $n/2$, it has to be connected to at least $n/2$ other vertices. This implies that in $|V(C_1)| = (n/2) + 1$. Similarly for component C_2. Each vertex $v \in C_2$ has degree at least $n/2$ and therefore $|V(C_2)| = (n/2) + 1$.

Number of vertices in graph G is sum of vertices in components C_1 and C_2.

Hence

$$|V(G)| = |V(G_1)| + |V(G_2)| = \frac{n}{2} + 1 + \frac{n}{2} + 1 = n + 2$$

which is contradiction since the graph G has n vertices. □

Dirac's theorem

Theorem 1. *Every graph with $|V| \geq 3$ and minimum degree at least $n/2$ has a Hamilton cycle.*

Proof. Let $P = x_0x_1...x_k$ be the longest path in G. By maximality of P all neighbors of x_0 and x_k have to lay in a path, otherwise P would not be the longest path. Therefore $n/2$ neighbors of x_0 and $n/2$ neighbors of x_k are lying on the path. By the pigeonhole principle, there must exist an edge $x_0x_{i+1} \in E$ and $x_ix_k \in E$. Then the Hamiltonian cycle is formed as

$$C = x_0x_{i+1}x_{i+2}\ldots x_{k-1}x_kx_ix_{i-1}\ldots x_1x_0$$

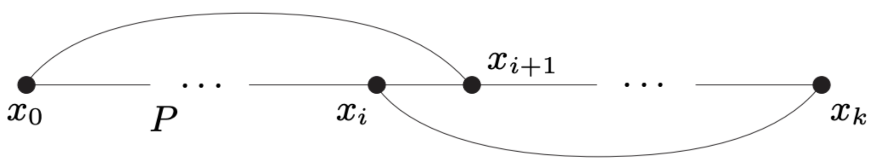


Figure 2.2: Hamilton cycle in a proof of Dirac's Theorem

If this is not a Hamiltonian cycle ($k \neq n$), this would mean that there exists y such that $y \notin V(C)$. But since G is connected (see Lemma 1) there must exists an edge $yx_j \in E$ for some $0 \leq j \leq k$. But then a path $P' = yx_j, x_j + ...$ would be longer than our path P which would be in contradiction with our assumption. □

Ore's Theorem

Theorem 2. *Let G be a simple graph with $n \geq 3$ vertices. Suppose that for every pair of non-adjacent vertices u and v in G, we have*

$$\deg(u) + \deg(v) \geq n.$$

Then G is Hamiltonian.

Proof. Assume by contradiction that G satisfies the degree condition above, but is not Hamiltonian.

Now, construct a graph H by adding edges to G (without violating the degree condition) until no more edges can be added without creating a Hamiltonian cycle. So H is a maximal non-Hamiltonian graph that still satisfies the Ore condition.

This means that in H, for any two non-adjacent vertices u and v, adding the edge uv would result in a Hamiltonian cycle.

Let u and v be any such non-adjacent vertices in H. Since adding uv creates a Hamiltonian cycle, there must exist a Hamiltonian path between u and v in H. Let this path be

$$P = x_1, x_2, \ldots, x_n,$$

where $x_1 = u$ and $x_n = v$, and all x_i are distinct vertices of H.

Now, for each $2 \leq i \leq n$, observe the following: if both ux_i and vx_{i-1} are edges in H, then we could form a Hamiltonian cycle contradicting the assumption that H is not Hamiltonian.

Therefore, for each $2 \leq i \leq n$, at most one of the edges ux_i or vx_{i-1} can exist. This gives us at most $n - 1$ such edges involving u and v.

Hence,

$$\deg(u) + \deg(v) \leq n - 1,$$

which contradicts the assumption that $\deg(u) + \deg(v) \geq n$ for every pair of non-adjacent vertices.

Thus, our initial assumption must be false, and G must be Hamiltonian. □

2.2.2 Sufficient conditions for regular graphs

The requirement that the graph is regular introduces additional conditions for a graph to be Hamiltonian.

Erdős and Hobbs

In [9] Paul Erdős and Arthur Hobbs presented following theorem:

Theorem 3. *Let G be a 2-connected graph which is regular of degree $n - k$, where $k \geq 3$.*
If $|V(G)| = 2n$ and $n \geq k^2 + k + 1$, then G is hamiltonian.
If $|V(G)| = 2n - 1$ and $n \geq 2k^2 - 3k + 3$ then G is hamiltonian.

Proof to this theorem can be found in [9]

2.2.3 Jackson

Jackson in [16] provided proof of this theorem:

Theorem 4. *Every 2-connected, k-regular graph with $|V(G)| \leq 3k$ is hamiltonian.*

2.3 Algorithms

Finding the Hamiltonian cycle in a graph is very hard problem because we don't know the exact polynomial algorithm. Therefore we have to first examine if are sufficient conditions met and if not we have to use heuristics. In this chapter, we will provide a brief overview of heuristics algorithms for finding Hamiltonian cycles and one algorithm for the Traveling salesman problem since this algorithm will be used in this thesis.

2.3.1 Heuristic algorithms

Posa's Algorithm

Posa's in his work never described the algorithm for finding the Hamiltonian cycle, but his observation helped with another algorithm development. This algorithm does not use backtracking, rather it uses a technique called rotational transformation.[28]. The idea behind this approach is to use the so-called partial path and extend it, but if no vertex can be added to partial path end vertices, the order of the path is changed, so there will be a possibility of extending the partial path. The algorithm quits if all neighbors of endpoints are reordered such that they are and points and extensions of the partial path are not found.

MultiPath Algorithm

This algorithm although is backtracking it is not using a primitive approach. The idea behind this algorithm, besides an effective pruning technique, is building a cycle with multiple paths (segments) and if the paths can be merged together they are. At each step of the algorithm, the next extended endpoint of the segment is chosen randomly. This way is more effective since some edges are forced (they have to be in this particular

Hamilton cycle) or the other way edges that cannot be part of the Hamilton cycle are removed. [28]

2.3.2 Lin-Kernighan heuristic for solving the traveling salesman problem

Later in this thesis, we will use this algorithm, designed for solving the traveling salesman problem (further TSP), for solving the Hamilton cycle problem.

The idea of this algorithm is taken for k-opt algorithms, where we start with a valid TSP solution and we try to change k edges in such a way that we decrease the total value of TSP. But, main difference between k-opt is that in this algorithm the parameter k is not a constant but it is changing within an algorithm run. Algorithm starts with 2-opt and it increase a k only as long as changes improves a tour. This improvement is significant since it allows to escape local minima and therefore find a better solution. [21]

2.4 Complexity classes

Since we have mentioned that finding Hamiltonian cycles in a graph is an NP-Complete problem, it is essential to clarify and define three key complexity classes: P, NP, and NP-complete problems.

P Problems This class includes problems that can be solved in polynomial time. More specifically, the time complexity of such problems is expressed as $O(n^k)$ for some constant k, where n represents the size of the problem to be solved. [7]

NP Problems This class consists of problems that cannot be solved in polynomial time but can be verified in polynomial time. In other words, if a solution is provided, verifying whether it is correct can be done in polynomial time. It is important to note that every P problem is also an NP problem. [7]

NP Complete Problems This class is the most complex. It includes problems that belong to NP and have the additional property that any other NP problem can be reduced to them in polynomial time. In other words, if a polynomial-time algorithm were found for any NP-Complete problem, it would imply that every problem in the NP class could also be solved in polynomial time. [7]

NP-Hard Problems This class includes problems that are at least as hard as the hardest problems in NP, but they do not necessarily belong to the NP class themselves.

That means an NP-Hard problem may not even be a decision problem, and verifying a given solution might not be possible in polynomial time. All NP-Complete problems are also NP-Hard, but not all NP-Hard problems are NP-Complete.[7]

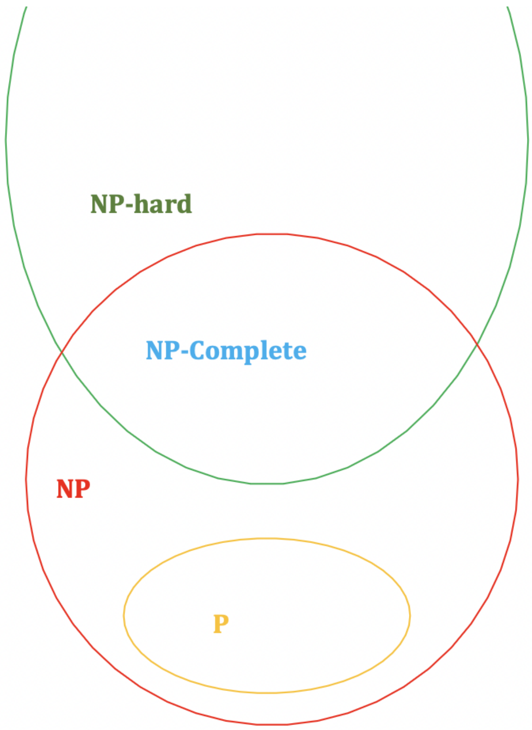


Figure 2.3: Complexity classes where $NP \neq P$

This leads us to one of the most fundamental and open questions in theoretical computer science: does the class NP equal the class P?

Chapter 3

Cages

This chapter provides an overview of the concept of cages, their mathematical definition, and known constructions. We will also explore record graphs, discuss Sachs's method used to construct Hamiltonian graphs with cage properties and highlight some of the open problems and challenges that remain in this area of research.

3.1 Definition

The problem of cages asks for finding construction of a simple regular graph G with specified degree and girth and minimal order.[11]

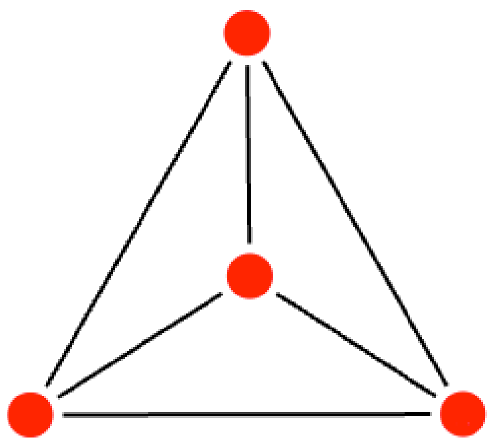


Figure 3.1: (3,3)-Cage

We denote graphs that satisfy the condition above as $n(k, g)$ where k is the regularity of the graph (1.1.1) and g is the girth of a graph (1.2). If the graph is a cage then we denote it by (k, g)-cage as you can see on 3.1 where is a cage with $k = 3$ and $g = 3$.

3.2 History and origin

Tutte first studied the origin of the problem and its variation where the condition of the cage is also to be Hamiltonian was studied by Kárteszi who provided a construction for graphs that are Hamiltonian and have $g = 6$ [19, 11].

3.2.1 Moore bound

Moore bound is the upper bound for graphs with the largest possible order (number of vertices) and maximum degree Δ and diameter d.[22]. The Moore bound is denoted as $M(\Delta, d)$.

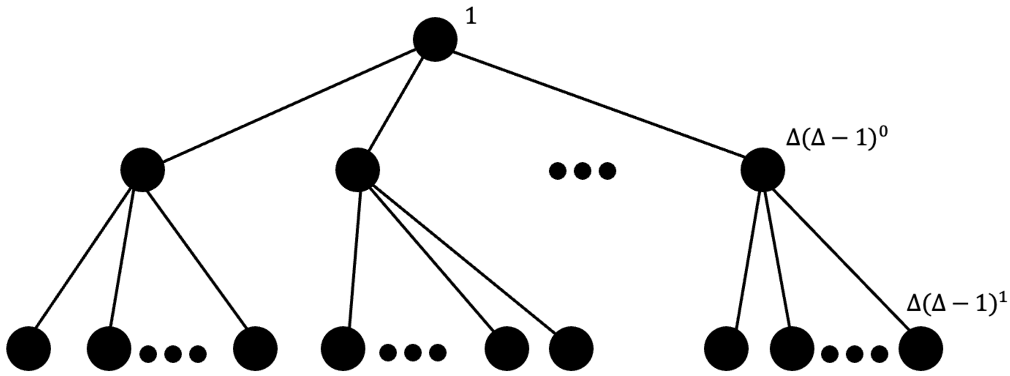


Figure 3.2: The idea behind deriving Moore Bound

By observation from 3.2, bound can be expressed as

$$1 + \Delta + \Delta(\Delta - 1) + \cdots + \Delta(\Delta - 1)^{d-1}$$

This formula represents the total number of vertices that can be reached with d steps from the starting vertex.

$$M(\Delta, d) = 1 + \Delta \sum_{i=0}^{d-1} d(d-1)^i$$

Δ	d	Moore Graph
≥ 2	1	Complete graphs $K_{\Delta+1}$
2	≥ 2	Cycles C_{2d+1}
3	2	Petersen graph
7	2	Hoffman-Singleton graph

Table 3.1: Known Moore graphs for given maximum degree Δ and diameter d.

Graphs which do satisfy this equality are rare and they are called Moore Graphs. Table 3.1 shows known Moore graphs for various parameters. Although Hoffman and Singleton proved that for $d = 2$ exists Moore graph with degree $\Delta = 2; 3; 7; 57$ and no others, example of $M(2, 57)$ was not found yet.[22]

Moore graphs and Cages have very close relationship. The girth of Moore graphs must be $g = 2d+1$. Therefore each Moore graph with $M(\Delta, 2g+1)$ is also $(k, g)-cage$ with $\Delta = k$. [11]

3.3 Known cages

For given parameters k and g, there can be more than one cage. In other words, multiple non-isomorphic graphs can exist with the same parameters and minimal order. This section contains table, where are listed known cages for parameters and also the number of cages and their order.

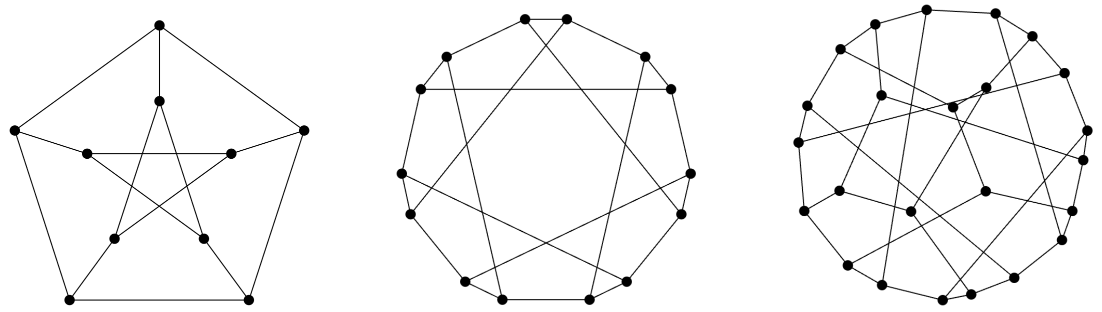
k	g	Name	# of Cages	Order
3	5	Petersen Graph	1	10
3	6	Heawood Graph	1	14
3	7	McGee Graph	1	24
3	8	Tutte-Coxeter Graph	1	30
3	9		18	58
3	10		3	70
3	11	Balaban Graph	1	112
3	12	Benson Graph	1	126
4	5	Robertson Graph	1	19
5	5		4	30
6	5		1	40
7	5	Hoffman-Singleton Graph	1	50
7	6		1	90
4	7		N/A	67

Table 3.2: Known cages for various values of degree k and girth g.

On 3.2 can be seen that number of vertices grows singificantly with the raising parameters k and g. This implies that problem of cages is not primitive.

It should be noted that cages are known for the parameters $g = 3$ and $k \geq 3$, as the corresponding graph is the complete graph K_{k+1}, and for $g = 4$ and $k \geq 3$, where the graph is the complete bipartite graph $K_{k,k}$.

Another important note to this topic is that cages are 2-connected.[24]



(a) Petersen Graph (b) Heawood Graph (c) McGee Graph

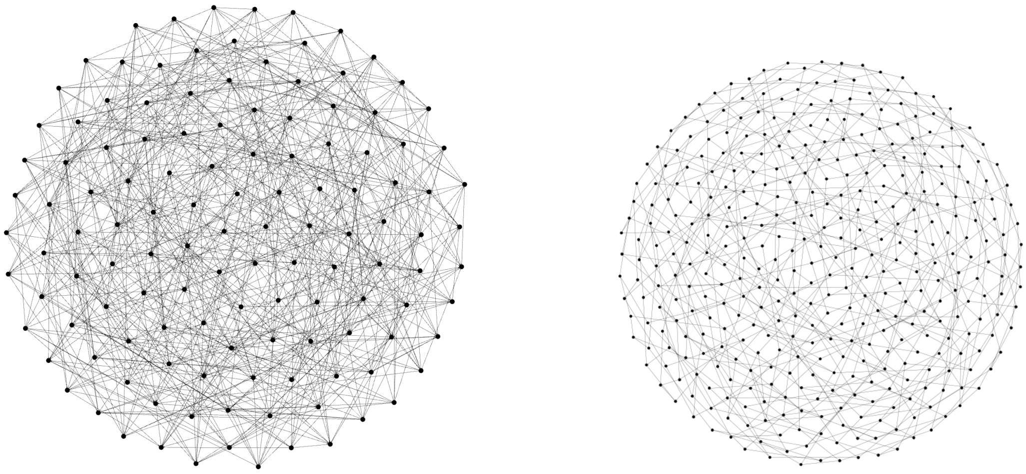
Figure 3.3: Cages with $k = 3$

3.4 Record graphs

In opposite of Moore graphs, Sachs in his work proved that there exists (k, g)-cage for any pair of (k, g), where $k \geq 3$ and $g \geq 2$.[26]. When order of (k, g)-cage is unknown we denote the smallest known k-regular graph with girth g as Record graph - rec(k, g).

k/g	5	6	7	8	9	10	11	12	13	14	15	16
3	10	14	24	30	58	70	112	126	272	384	620	960
4	19	26	67	80	275	384		728				
5	30	42	152	170		1296	2688	2730	7812			
6	40	62	294	312				32928				
7	50	90		672				39216				
8	80	114	1152	800			74752	74898				
9	96	146		1170				132860				
10	124	182		1640				319440				
11	154	240		2618				354312				
12	203	266		2928				738192				
13	230	336		4342				804468				
14	288	366		4760				1957376				
15	312	462		7648				2088960				
16	336	504		8092				2236962				
17	448	546		8738				3017196				
18	480	614		10440				4938480				
19	512	720		13642				5227320				
20	576	762		14480								

Table 3.3: Summary of upper bounds for $n(k, g)$. [11]



(a) $rec(10, 5)$ (b) $rec(3, 14)$

Figure 3.4: Record graphs

3.5 Construction of regular hamiltonian graphs with given girth

Sachs in [26] provided proof that for each parameter $k \geq 3$ and $g \geq 2$ there exists a graph which girth is g, regularity is k and graph is Hamiltonian. He also proved more properties but they are beyond this thesis and therefore we will not consider it in this chapter.

3.5.1 Construction

The construction is divided into two cases, based on the girth of the graph.

Case A: Girth $g = 2$

- **Base Case:** For $g = 2$, we begin with the base case of a graph consisting of two vertices, u and w, connected by two parallel edges, e_1 and e_2. This graph is trivially Hamiltonian.
- **Inductive Step:** For $k > 3$, we construct a graph G with girth 2 as follows:
  1. Let $V_1 = \{v_1, v_2, \ldots, v_{k-1}\}$ and $V_2 = \{v'_1, v'_2, \ldots, v'_{k-1}\}$ be two disjoint sets of $k - 1$ vertices each.
  2. Construct the complete bipartite graph $H_{k-1} = K_{k-1,k-1}$. This graph is formed by connecting each vertex in V_1 to each vertex in V_2 with a single edge.
  3. The graph H_{k-1} is a $(k-1)$-regular bipartite graph. It contains a Hamiltonian cycle $C = (v_1, v'_1, v_2, v'_2, \ldots, v_{k-1}, v'_{k-1}, v_1)$.

4. By König [20], the bipartite graph H_{k-1} can be decomposed into $k-1$ 1-factors. Select one of these 1-factors and each edge e replace with a double edge (two parallel edges). Let the resulting graph be G.
5. The resulting graph G is Hamiltonian (since the original Hamiltonian cycle in H_{k-1} remains valid) and has girth 2, due to the presence of the double edges.

Remark. The construction in Case A yields a graph with $2(k-1)$ vertices. It is noted that for some values of k, graphs satisfying the same properties with fewer vertices may exist.

Case B: Girth $g_0 \geq 3$

We extend the construction to graphs with girth $g_0 \geq 3$ using a double inductive approach.

- **Inductive Hypothesis on Girth:** Assume that for all $g = 2, 3, \ldots, g_0 - 1$ (where $g_0 \geq 3$) and all $k \geq 2$, there exists a Hamiltonian graph with girth g. We now construct a graph with girth g_0 and regularity k.

- **Base Case for Induction on Regularity ($k = 2$):** A cycle C_{g_0} of length g_0 forms a 2-regular graph with girth g_0 and is Hamiltonian.

- **Inductive Step on Regularity:** Assume that for a fixed girth g_0, graphs with regularity $k = 2, 3, \ldots, k_0 - 1$ are Hamiltonian and have girth g_0. We construct a graph with girth g_0 and regularity k_0 as follows:

  1. By the inductive hypothesis, there exists a graph Δ with girth g_0 and regularity $k_0 - 1$. Let the vertices of a Hamiltonian cycle $S(\Delta)$ of Δ in cyclic order be $d_1, d_2, \ldots, d_l$.
  2. By the inductive hypothesis on girth, there exists a graph Λ with girth $g_0 - 1$ and regularity l. Let the vertices of a Hamiltonian cycle $H(\Lambda)$ of Λ in cyclic order be $v_1, v_2, \ldots, v_m$.
  3. Construct m mutually disjoint graphs $\Delta_1, \Delta_2, \ldots, \Delta_m$, each isomorphic to Δ. For each $i = 1, \ldots, m$, let $d_{i1}, d_{i2}, \ldots, d_{il}$ be the vertices of Δ_i, such that there is an isomorphism between Δ and Δ_i where d_j in Δ corresponds to d_{ij} in Δ_i for all $j = 1, \ldots, l$. Then, $d_{i1}, d_{i2}, \ldots, d_{il}$ form a Hamiltonian cycle $E(\Delta_i)$ in cyclic order for each $i = 1, 2 \ldots m$.
  4. Construct a graph F with k_0 and girth g_0 as follows:

(a) **Step 1: Substitution.** Substitute the graph Δ_i for each vertex v_i in Λ. This means that where v_i was in Λ, we now place a copy of the graph Δ (which we are calling Δ_i).

(b) **Step 2: Connecting along $H(\Lambda)$.** For each edge (v_i, v_{i+1}) in the Hamiltonian cycle $H(\Lambda)$ of Λ (with indices taken modulo l), add an edge between the vertex d_{il} in Δ_i and the vertex $d_{(i+1)1}$ in Δ_{i+1}. This step connects the graphs Δ_i together in a chain, following the Hamiltonian circuit of Λ. This creates a Hamiltonian circuit in F: $d_{11}, d_{12}, \ldots, d_{1l}, d_{21}, d_{22}, \ldots, d_{2l}, d_{31} \ldots, d_{m-1l}, d_{m1}, d_{m2} \ldots, d_{ml}$.

(c) **Step 3: Connecting remaining edges.** Let $\Delta_i' = \Delta_i - \{d_{i1}, d_{il}\}$ for $i = 1, \ldots, m$. For each edge (v_i, v_j) in Λ that is not part of the Hamiltonian circuit $H(\Lambda)$, add an edge between a vertex in Δ_i' and a vertex in Δ_j'. The edges are added in such a way that each vertex in $\Delta_1', \ldots, \Delta_m'$ is incident to exactly one such edge.

This constructive proof by double induction demonstrates a method to create Hamiltonian graphs for all $k \geq 2$ and $g \geq 3$.

3.5.2 Algorithm

Althought construction proof is complex and hard to understand the algorithm is precise and understandable.

The algorithm proceeds by induction on k and g, starting with two base cases.

Base Case: $k = 2$ and $g \geq 3$

Construct a simple cycle of length g. This graph is trivially Hamiltonian.

Base Case: $k \geq 2$ and $g = 2$

Construct a multigraph consisting of two vertices, u and w, connected by k edges. This graph is also trivially Hamiltonian.

The inductive step reduces the problem by constructing two simpler graphs:

- One with degree $k - 1$ and the same girth g,
- Another with the same degree as the number of vertices in the first graph, and girth $g - 1$.

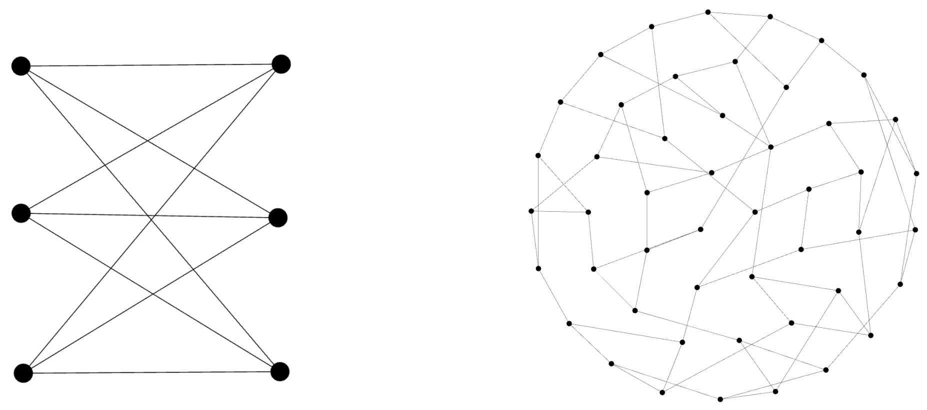
The final graph is then built using the ReplaceVerticesWithGraph procedure, ensuring that the resulting graph maintains the desired properties of regularity and girth.

Algorithm 1 Sachs Construction

Procedure SachsConstruction(k, g)
if $g = 2$ **then**
    return $G \leftarrow$ multigraph with two vertices and k edges
end if
if $k = 2$ **then**
    return $G \leftarrow$ cycle graph of length g
end if
$G_\Delta \leftarrow$ SachsConstruction$(k-1, g)$
$G_\Lambda \leftarrow$ SachsConstruction$(|V(G_\Delta)|, g-1)$
$G \leftarrow$ ReplaceVerticesWithGraph(G_Δ, G_Λ)
return G

Algorithm 2 Replace Vertices With Graph

Procedure ReplaceVerticesWithGraph(Δ, Λ)
$F \leftarrow$ empty graph
for $i = 1$ to λ **do** $\triangleright$ $\lambda = |V(\Lambda)|$
    Create disjoint copy Δ_i of Δ
    Relabel vertices of Δ_i uniquely
    Add Δ_i to F
end for
for $i = 1$ to λ **do**
    Connect a fixed endpoint of Δ_i to the next $\Delta_{i+1 \pmod{\lambda}}$ along the cycle
end for
$\triangleright$ $H(\Lambda)$ - edges of Hamiltonian cycle in Λ
for each remaining edge $(v_i, v_j) \in E(\Lambda) \setminus H(\Lambda)$ **do**
    Select an unused vertex u from Δ_i
    Select an unused vertex w from Δ_j
    Add (u, v) to $E(F)$
end for
return F



(a) $(3, 4)$-cage (b) Hamiltonian $(3, 4)$ graph by Sachs

Figure 3.5: Comparison of the Orders of (3,4)-cage and Sachs's Construction

From Figure 3.5, it can be seen that the graphs generated by Sachs's construction exhibit a significantly higher order than cages. The order N of $n(k, g)$ generated from construction is

$$N(k, g) = N(k-1, g) \cdot N(N(k-1, g), g-1)$$

which is why these graphs cannot be practically generated since their size and recurrence in their construction is very time and memory consuming. Table 3.4 shows difference of order between constructed graphs and cages.

k	g	Cage order	Construction order
3	3	4	6
3	4	6	48
4	3	5	12
4	4	8	10133099161583616
5	3	6	24
6	3	7	48
7	3	8	96
8	3	9	192

Table 3.4: Order differences

From the order differences shown in Table 3.4, it can be observed that graphs generated by Sachs's constructions have relatively small orders when the girth is 3. However, for larger girths, the order of the graphs increases significantly.

Our implementation of this construction together with a method for determining the orders of the corresponding graphs, can be found on [27].

Chapter 4

System Usage

In this chapter, we present the methods and software tools utilized in our thesis for testing Hamiltonicity and for graph visualization. We also discuss their respective advantages and limitations.

4.1 Hamiltonicity finding software

Besides heuristics or precise backtracking algorithms, there is no software that specializes in finding Hamiltonian cycle in a graph. The majority of software in this field is focused on solving the Traveling salesman problem 1.1.1. Therefore before we used tools specified in this section, we had to reduce finding Hamiltonian cycle to problem of finding optimal solution for complete TSP. This reduction is in more detail specified in next chapter.

4.1.1 OR-Tools

OR-Tools is an open-source software suite developed by Google for solving various optimization problems, including linear programming, constraint programming, and combinatorial optimization. In this thesis, OR-Tools was used with different search parameters to explore small-order cages.

To solve our instances, we followed the general approach outlined in the official OR-Tools documentation [8]:

1. **Define the data model**: We specify the number of vertices and construct a distance matrix representing the distances between each pair of vertices. The method used to generate this matrix is described in Section 5.2.

2. **Create the routing index manager**: This component manages the conversion between the external vertex numbering and the internal indexing system used by OR-Tools.

3. **Create the routing model**: This model encapsulates the problem to be solved.

4. **Define the cost function**: In our case, the cost function corresponds to the edge weights between connected vertices.

5. **Set search parameters**: We configure parameters such as the first solution strategy and the local search metaheuristics to guide the solver.

6. **Start the solver**: Solve the problem and retrieve the solution.

The main advantage of using OR-Tools is its well-written and comprehensive documentation, which makes it easier to understand the available tools and methods. However, the wide range of configuration options - especially when setting search parameters — can also make it more challenging to use effectively, often requiring experimentation with different approaches to achieve optimal results.

The modified version of the original example code used in our experiments is available on GitHub [27].

4.1.2 Concorde

The main software used in our work is Concorde. Concorde is a highly efficient and widely used tool for solving the TSP and several related combinatorial optimization problems. One of its benefits is its ability to compute optimal solutions to TSP instances, making it a standard tool in the field of combinatorial optimization.[1, 3]

In addition to finding exact solutions, Concorde also includes implementations of heuristic algorithms as the Lin-Kernighan heuristic (see 2.3.2). This heuristic is particularly useful for tackling large instances of the TSP where obtaining an exact solution would be computationally ineffitiens or not possible.

Concorde's optimal solving capabilities rely on the linear programming solver QSopt, which is specifically designed to be easily integrated into other software systems.[2] QSopt provides robust and efficient LP-solving functionality, which is critical for the branch-and-cut algorithms used by Concorde to solve the TSP to optimality. Together, Concorde and QSopt form a powerful framework for addressing complex instances of TSP and related problems.

Concorde is solving TSP for complete graphs, therefore we need to transform $n(k, g)$ to be complete. This trasnformation is further described in Section 5.2.

Graphs tested in Concorde must be represented in the TSPLIB format for TSP, which is a format used for problems related to the Traveling Salesman Problem. Although TSPLIB also supports the HCP type for the Hamiltonian Cycle Problem, Concorde is a TSP solver, so it is necessary to use the correct type. On 4.1 can be seen

```
NAME: 3 - 5_TSP
TYPE: TSP
COMMENT: Converted from adjacency list to TSP instance
DIMENSION: 10
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
EDGE_WEIGHT_SECTION
0 1 1 1 10000000000 10000000000 10000000000 10000000000 10000000000 10000000000
1 0 10000000000 10000000000 1 1 10000000000 10000000000 10000000000 10000000000
1 10000000000 0 10000000000 10000000000 10000000000 1 10000000000 10000000000 1
1 10000000000 10000000000 0 10000000000 10000000000 10000000000 1 1 10000000000
10000000000 1 10000000000 10000000000 0 10000000000 1 10000000000 1 10000000000
10000000000 1 10000000000 10000000000 10000000000 0 10000000000 1 10000000000 1
10000000000 10000000000 1 10000000000 1 10000000000 0 1 10000000000 10000000000
10000000000 10000000000 10000000000 1 10000000000 1 1 0 10000000000 10000000000
10000000000 10000000000 10000000000 1 1 10000000000 10000000000 10000000000 0 1
10000000000 10000000000 1 10000000000 10000000000 1 10000000000 10000000000 1 0
EOF
```

Figure 4.1: Petersen graph in TSPLIB format

Peteresen graph represented as TSP in TSPLIB format. Graphs in such representation has to be saved as `.tsp`. [25]

4.2 Graph Visualization

Graph visualization is an important tool for better understanding the structure and properties of graphs. It helps to quickly identify key features like clusters, important vertices, and connectivity patterns. In case of cages and record graph is interesting its level of symmetry which can be observed on visualisations.

There are many tools available for graph visualization, but for our work, we mainly used Gephi.

4.2.1 Gephi

Gephi is an open-source software for visualizing and analyzing graphs and networks. [4] It provides a user-friendly interface, allowing users to load graph data, apply layouts, and explore different properties of the graph, such as degree distribution, connected components, or modularity.

Gephi supports various file formats, which makes it flexible for different types of projects. It also offers a range of layout algorithms, including ForceAtlas2, which are useful for creating clear and informative visual representations of complex networks. In this thesis, we used Gephi mainly to visualize record graphs for promoting this topic and for presenting complete datasets.

4.3 Devana

Devana is a Slovak high-performance computer (HPC) used for computations that would be time-consuming on regular computers. It is mainly used for scientific research. [6] The main advantage of using this HPC is the large number of CPUs. This means that problems which can be parallelized are very efficient, as they can be distributed across multiple CPUs, increasing effectiveness and decreasing the time needed to solve the problem. In this work, we used Devana to find Hamiltonian cycles in large instances of record graphs, namely: $Rec(3, 18)$ and $Rec(3, 20)$, which would not be feasible on regular computers.

Before the experiments, Devana did not have the Concorde module installed. It had to be initialized and configured for this work.

Chapter 5

Experiment

This chapter presents our work on testing Hamiltonicity in regular graphs. It covers how the graphs were obtained, their representations and transformations, the details of the experiment, and the availability of the datasets.

5.1 Datasets

In this section are described sources from where we obtained tested cages and record graphs. Whole datasets can be found and downloadable from

Citation formatted zenodo

5.1.1 Cages Datasets

Cages are well-known, relatively small graphs. For this experiment, we used a collection of cage graphs presented in [11]. Their explicit representations are available on the House of Graphs website [23]. We used the adjacency list representation, which can be downloaded in the `.lst` format from the site.

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

Figure 5.1: Adjacency list representation of Petersen graph taken from [23]

Each line of the `.lst` file describes the adjacency information for one vertex. The number before the colon denotes the vertex label, and the numbers after the colon represent its adjacent vertices 1.3.2. In House of Graphs vertex labeling starts with 1.

5.1.2 Record graphs datasets

Record graphs are typically not represented explicitly. Due to their large order, they are usually described through a construction. A detailed explanation of the construction methods for various record graphs is beyond this thesis. Thanks to Geoffrey Exoo, many record graphs are available on his website [10], where they are provided in the following format:

```
115  282  126
142  267  118
118  275  136
128  259  137
129  272  112
130  287  100
137  270  129
123  279  131
135  280  105
...
```

Figure 5.2: Beginning of rec(3, 14) represented as adjacency list obtained from [10].

Form of adjacency list is different from one in 5.1. The source vertex is represented as line number starting with 0. Therefore first line (115 282 126) represents edges $E = \{(0, 115), (0, 282), (0, 126)\}$.

5.2 Data transformations

In this section, we will describe manipulation with graphs and their representations. We will define the format in which the Hamiltonian cycle is represented. All scripts necessary for these transformations are available on [27]

5.2.1 Inputted Data format

Graphs in their representations obtained from [23, 10] we transform to united `.lst` format. Cages were already in this format, therefore, the migration was applied to record graphs. Migrated 5.9 graph to `.lst` format is on next image:

```
0: 115  282  126
1: 142  267  118
2: 118  275  136
3: 128  259  137
4: 129  272  112
5: 130  287  100
6: 137  270  129
7: 123  279  131
8: 135  280  105
...
```

Figure 5.3: Beginning of rec$(3, 14)$ represented as adjacency list saved in `.lst` format.

We left the convention that the first line source vertex is labeled as 0. This is due to easy manipulation, although it brings non-consistency with cage representations since for these graphs, labeling starts with 1.

5.2.2 Reduction of HCP to TSP

Since Concorde is a complete TSP solver, which means that the input must contain the distance between each pair of vertices. Therefore, it does not support solving HCP. The reduction was necessary to run the experiment using Concorde.

Reduction was done by following the algorithm:

Algorithm 3 Reduction of HCP to TSP

Require: A graph $G = (V, E)$ for the HCP
Ensure: A complete weighted graph $G' = (V, E')$ for the TSP

Let $n \leftarrow |V|$
Initialize G' as a complete graph with vertex set V and distances equal to 0
for all pairs of vertices (u, v) where $u \neq v$ **do**
    if $(u, v) \in E$ **then**
        Set weight $w(u, v) \leftarrow 1$
    else
        Set weight $w(u, v) \leftarrow 10000$
    end if
end for
return G' with edge weights w

The resulting representation is therefore an adjacency matrix A with following properties:

$$A[u,v] = \begin{cases} 0 & \text{if } u = v \\ 1 & \text{if } (u,v) \in E(G) \\ 10000 & \text{if } (u,v) \notin E(G) \end{cases}$$

This will ensure that graph is complete. Hamilton cycle is found iff the resulting cycle has distance same as number of vertices. In other cases, cycle contain edge which is not in initial graph G.

5.2.3 TSPLIB format - input to Concorde

As mention in 4.1.2, Concorde accepts graphs in TSPLIB format and TSP instance. Therefore files has to be saved as `.tsp`. The structure of file is defined as

```
NAME: < Name for identification >
TYPE: < Various choices in this thesis TSP >
COMMENT:
DIMENSION: < Number of vertices of graph >
EDGE_WEIGHT_TYPE: < various types in this thesis EXPLICIT >
EDGE_WEIGHT_FORMAT: < various types in his thesis FULL_MATRIX >
EDGE_WEIGHT_SECTION
...
< Matrix of reduced graph >
...
EOF
```

Figure 5.4: Structure of TSPLIB format

More information about the parameters of the TSPLIB format can be found in [25]. We used the naming convention `k - g_TSP`, where the name reflects the parameters of the graph. The comment field was set to `Converted from adjacency list to TSP instance`.

5.2.4 Cycle Representation – Output from Concorde

After running Concorde, we obtained a cycle in the form of a file that represents the solution to TSP.

The cycle is defined explicitly as shown on 5.5. The first line indicates the number of vertices in the graph. Each subsequent line represents an edge in the cycle and is written in the format

$$v_i \quad v_{i+1} \quad w(v_i, v_{i+1})$$

where:

- v_i is the current vertex in the cycle,
- v_{i+1} is the next vertex in the cycle, and
- $w(v_i, v_{i+1})$ is the weight of the edge connecting v_i and v_{i+1}.

```
10 10
0 1 1
1 5 1
5 9 1
9 7 1410065408
7 3 1
3 8 1
8 4 1
4 6 1
6 2 1
2 0 1
```

Figure 5.5: TSP solution found by Concorde in the Petersen graph

This format describes the ordered sequence of edges that form a Hamiltonian cycle as computed by Concorde. The third column in the file can take one of two values:

- If it is set to 1, the edge exists in the original input graph.
- If it is set to 1410065408, the edge does not exist in the original graph but comes from the reduced complete graph. This large value is the highest possible default weight used during computation which Concorde supports.

5.3 Experimental Procedure

In this section, we provide details about the experiment as well as its results.

5.3.1 Verifying Sufficient Conditions

Since cages are sparse graphs, the sufficient conditions commonly used for dense graphs (Theorems 2.2.1 and 2.2.1) are generally not applicable. For a (k, g)-graph to be Hamiltonian under these theorems, the following inequality must hold:

$$n(k, g) \leq 2k.$$

Without loss of generality, we may substitute $n(k,g)$ with its known lower bound, the Moore bound $M(k,g)$ (see 3.2.1), because every (k,g)-graph must have at least $M(k,g)$ vertices. Therefore, if the inequality is not satisfied for the Moore bound, it certainly cannot hold for any (k,g)-graph with more vertices. Under this substitution, the inequality becomes:

$$M(k,g) = \begin{cases} 1 + k \cdot \sum_{i=0}^{\frac{g-3}{2}} (k-1)^i \leq 2k, & \text{if } g \text{ is odd,} \\ 2 \cdot \sum_{i=0}^{\frac{g-2}{2}} (k-1)^i \leq 2k, & \text{if } g \text{ is even.} \end{cases}$$

Even if this condition is met, it does not imply that a (k,g)-cage is Hamiltonian, since $n(k,g) \geq M(k,g)$, and the difference between the actual order and the Moore bound can be significant.

However, this condition is rarely satisfied for most combinations of k and g, which means that the Dirac and Ore conditions are generally unsuitable for the study of cages.

Since cages are 2-connected [24], the theorem provided by Jackson (see 2.2.3) might appear more suitable. According to his result, a 2-connected k-regular graph with at most $3k$ vertices is Hamiltonian, which allows us to restate the Hamiltonicity condition as

$$M(k,g) \leq 3k.$$

However, this bound is still not enough for cages, because the number of vertices in most (k,g)-cages is much larger than $3k$. As a result, Jackson's theorem doesn't generally apply to cages.

The theorem stated in 2.2.2 applies to graphs with high degree, which implies they are dense. In contrast, cages are sparse graphs with relatively low degrees, making this theorem unsuitable for studying their Hamiltonicity.

Additionally, we tested these conditions on various cages and regular graphs. The results are presented in Appendix 5.5.

5.3.2 Testing Hamiltonicity in Cages

To test Hamiltonicity in cages, we used the optimal solver implemented in Concorde. Each graph was tested only one time. The results are summarized in the following table. In cases where multiple cages exist for the same parameters, all of them yielded identical results as noted. After 20 runs, we observed that all graphs were Hamiltonian, except for the Petersen graph, which is the (3,5)-cage.

k	g	Order	Hamiltonian
3	5	10	✗
3	6	14	✓
3	7	24	✓
3	8	30	✓
3	9	58	✓
3	10	70	✓
3	11	112	✓
3	12	126	✓
4	5	19	✓
5	5	30	✓
6	5	40	✓
7	5	50	✓
7	6	90	✓
4	7	67	✓

Table 5.1: Tested cages with Hamiltonian cycle findings.

The non-Hamiltonicity of the Petersen graph was expected, as it is proven in [15] that the Petersen graph does not contain a Hamiltonian cycle. We focused on the smallest $(3, 5)$-graphs that could potentially be Hamiltonian. In particular, [17, 18] presents two $(3, 5)$-graphs with excess 2, each consisting of 12 vertices.

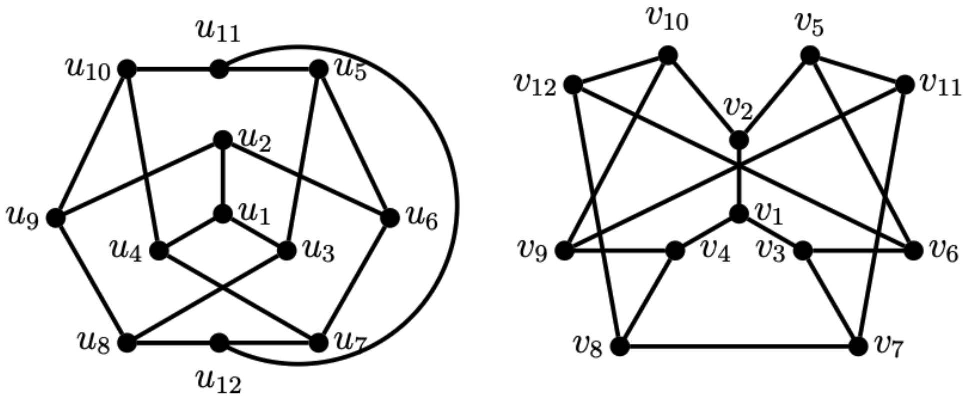


Figure 5.6: The two $(3, 5)$-graphs with excess 2

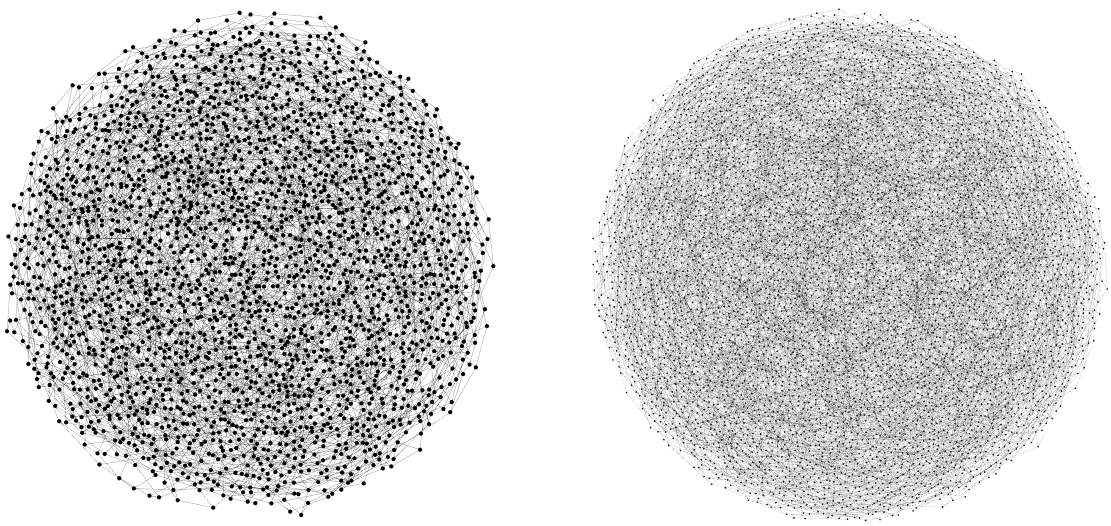
Since the Hamiltonicity of these graphs was not known in advance, we tested them using the Concorde TSP solver. The results show that both graphs are Hamiltonian.

Therefore, the two smallest known Hamiltonian $(3, 5)$-graphs are those shown in Figure 5.6.

5.3.3 Testing Hamiltonicity in Record graphs

Since record graphs have significantly higher order compared to cages, using an exact solver was not feasible in this case. All graphs listed in the table were tested 20 times using the Lin-Kernighan heuristic method (see Section 2.3.2), which is also implemented in Concorde. The main disadvantage of this heuristic is that failure to find a Hamiltonian cycle does not imply that such a cycle does not exist. This issue arose in the cases of the graphs rec$(3, 18)$ and rec$(3, 20)$, which, due to their large order, were found to be non-Hamiltonian by the heuristic.

Therefore, the number of search steps was increased from the default $s = n(k, g)$ to $s = 100000$. However, this increase did not result in the discovery of a Hamiltonian cycle in either graph.



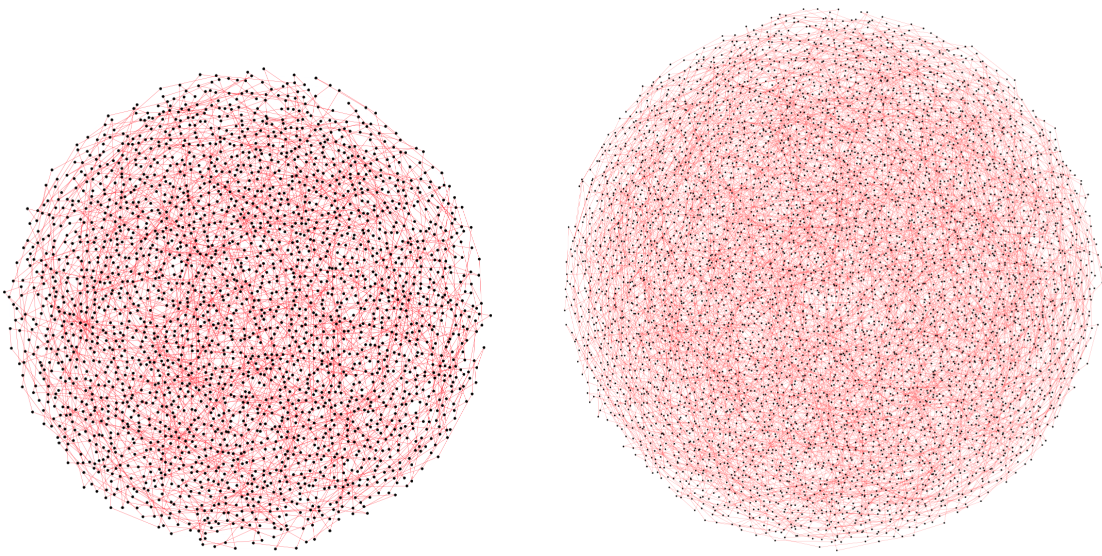
(a) Rec(3,18) (b) Rec(3,20)

Figure 5.7: Problematic record graphs

Hence, it was necessary to explore alternative approaches. We used Devana (see Section 4.3) to test rec$(3, 18)$ and rec$(3, 20)$ with an optimal algorithm. This allowed us to find Hamiltonian cycles in both graphs. As shown in Table 5.2, both graphs are Hamiltonian, and an example of a Hamiltonian cycle is shown in Figure 5.8.

k	g	Order	Hamiltonian
3	14	384	✓
3	16	960	✓
3	17	2176	✓
3	18	2560	✓
3	20	5376	✓
4	9	275	✓
4	10	384	✓
5	10	1296	✓
7	8	672	✓
10	5	124	✓
12	5	203	✓

Table 5.2: Tested record graphs with Hamiltonian cycle findings.



(a) HC in rec$(3, 18)$ (b) HC in rec$(3, 20)$

Figure 5.8: Hamiltonian cycle in problematic record graphs

In the end, all tested Record graphs were found to be Hamiltonian, including the difficult cases of rec$(3, 18)$ and rec$(3, 20)$. While the Lin-Kernighan heuristic was useful for most record graphs, it struggled with the larger ones. Using Devana with an exact algorithm helped confirm that these graphs do contain Hamiltonian cycles. These results support the idea that Record graphs, even as they get larger, tend to have Hamiltonian cycles.

5.4 Limitations

Since testing Hamiltonicity is an NP-complete problem (see 2.1), the test sets we used were subject to several constraints.

We encountered two main types of limitations:

- Graph transformation limitations
- Concorde memory constraints

5.4.1 Graph Transformation Limitations

For large graphs with more than 100000 vertices, we encountered difficulties during the transformation from the `.lst` format to the `.tsp` format. Due to the massive size of the graphs and their corresponding adjacency matrices, the conversion process became infeasible.

5.4.2 Concorde Memory Constraints

When running Concorde on the graph rec$(3, 23)$, which has 49326 vertices, on the machine Devana, we received the following error message:

```
Out of memory. Asked for 571239908 bytes
could not read the TSPLIB file
```

Figure 5.9: Concorde memory error on rec$(3, 23)$ using Devana.

This memory limitation arises from the large size of the TSPLIB file, which contains the full adjacency matrix. As a result, we were unable to proceed with testing even larger graphs.

5.5 Reproducibility and Data Access

As mentioned throughout this work, all code and datasets are freely available online.

- The source codes of Sachs' construction and helper functions for data transformations are available at [27].
- Datasets of cages and record graphs as adjacency matrices are available at...
- Datasets of Concorde input files are available at
- Concorde output results are available at .

odo or
reposi-

It should be noted that the HCP and TSP are NP-complete problems. Therefore using other computer may not find a solution.

Todo list

Conclusion

In this thesis, we focused on the existence of Hamiltonian cycles in (k, g)-cages—k-regular graphs of girth g and minimal order—as well as in $rec(k, g)$ graphs, the smallest known k-regular graphs with girth g. First, we obtained known cages from the House of Graphs database, where many were marked as Hamiltonian, although explicit Hamiltonian cycles were not provided. For record graphs, our primary source was Geoffrey Exoo's collection. To test for Hamiltonicity, we first checked classical sufficient conditions such as Dirac's and Ore's theorems, along with results specific to regular graphs from Jackson and Erdős–Hobbs. None of these conditions were satisfied by the cages or record graphs. Therefore, we applied the Concorde TSP Solver, transforming the Hamiltonian cycle problem into a Travelling Salesman Problem to verify Hamiltonicity computationally. Additionally, we developed a script to construct Hamiltonian (k, g)-graphs based on Sachs' method, and we compared the resulting graph orders to those of actual cages. This construction typically produced graphs of significantly higher order. Due to the time complexity of the algorithm, it was infeasible to complete computations for parameters with $g \geq 5$ and $k \geq 4$.

Our results showed that all known cages, except for the Petersen graph (the $(3, 5)$-cage), contain Hamiltonian cycles. In the case of record graphs, our analysis was limited by their increasing size; however, all tested instances were also found to be Hamiltonian. Based on these findings, we support the hypothesis:

For all (k, g)-cages, the graphs are Hamiltonian except for the unique case of the $(3, 5)$-cage.

If we apply this hypothesis to record graphs, we can say even more:

If a $rec(k, g)$ graph is not Hamiltonian, then there must exist a smaller k-regular graph with girth g that is Hamiltonian.

This would contradict the idea that the record graph is the smallest known graph with these properties. For this reason, it would be interesting to study such cases further and search for smaller graphs, as they might be even closer in size to the actual cages.

In the future, we plan to test record graphs from various sources. Since record graphs are often defined by construction methods and exact graph representations are not always available, we have recently contacted the authors of these graphs. Once we receive their responses, we will proceed with testing their graphs. Additionally, we aim to study how many Hamiltonian cycles cages or record graphs have for larger values of k and g. Another interesting problem to explore is proving that cages with even girth are bipartite.

This study makes progress in understanding the Hamiltonian properties of cages, showing both what has been achieved and what still needs to be solved for future research.

Bibliography

[1] David Applegate, Robert Bixby, Václav Chvátal, and William Cook. Concorde tsp solver. https://www.math.uwaterloo.ca/tsp/concorde/index.html. Accessed: 2025-04-26.

[2] David Applegate, William Cook, Sanjeeb Dash, and Monika Mevenkamp. Qsopt linear programming solver. https://www.math.uwaterloo.ca/~bico/qsopt/index.html. Accessed: 2025-04-26.

[3] David L. Applegate, Robert E. Bixby, Václav Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton and Oxford, 2006.

[4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. https://gephi.org, 2009. Accessed: 2025-04-26.

[5] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[6] Computing Centre of the Slovak Academy of Sciences. Supercomputer devana. https://vs.sav.sk/en/services/supercomputer-devana/, 2024. Accessed: 2025-04-26.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[8] Google Developers. Traveling salesperson problem | or-tools, 2024. Accessed: 2025-04-23.

[9] PAUL ERDŐS and Arthur M Hobbs. Hamiltonian cycles in regular graphs of moderate degree. *Journal of Combinatorial Theory, Series B*, 23:139–142, 1977.

[10] Geoffrey Exoo. Cages – a collection of record graphs. https://isu.indstate.edu/~gexoo/CAGES/, n.d. Accessed: 2025-04-29.

[11] Geoffrey Exoo and Robert Jajcay. Dynamic cage survey. *The electronic journal of combinatorics*, pages DS16–Jul, 2012.

[12] Reza Zanjirani Farahani. *Graph Theory for Operations Research and Management: Applications in Industrial Engineering: Applications in Industrial Engineering*. IGI Global, 2012.

[13] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Springer, New York, 2001.

[14] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics: An Applied Introduction*. Addison-Wesley, 5th edition, 1998.

[15] D. A. Holton and J. Sheehan. *The Petersen Graph*. Cambridge University Press, 1993.

[16] Bill Jackson. Hamilton cycles in regular 2-connected graphs. *Journal of Combinatorial Theory, Series B*, 29(1):27–46, 1980.

[17] Tatiana Jajcayová, Slobodan Filipovski, and Robert Jajcay. Counting cycles in graphs with small excess. pages 17–36, 01 2017.

[18] Tatiana Baginová Jajcayová, Slobodan Filipovski, and Robert Jajcay. Improved lower bounds for the orders of even girth cages. *The Electronic Journal of Combinatorics*, 23(3):P3.21, 2016.

[19] Francesco Kárteszi. Piani finiti ciclici come risoluzioni di un certo problema di minimo. *Bollettino dell'Unione Matematica Italiana, Serie 3*, 15(4):522–528, 1960. Digitized by the Biblioteca Digitale Italiana di Matematica (BDIM).

[20] Dénes Konig. *Theorie der endlichen und unendlichen Graphen*, volume 72. American Mathematical Soc., 2001.

[21] Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[22] Mirka Miller and Jozef Sirán. Moore graphs and beyond: A survey of the degree/-diameter problem. *The electronic journal of combinatorics*, pages DS14–May, 2012.

[23] House of Graphs. House of graphs. `https://houseofgraphs.org/`, n.d. Accessed: 2025-03-15.

[24] Tomaž Pisanski and Brigitte Servatius. A survey of the graph theory of the symmetric group. *Journal of Graph Theory*, 24(2):187–204, 1997.

[25] Gerhard Reinelt. Tsplib95: Sample Solutions to the TSP and related problems. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf`, 1995. Accessed: 2025-04-26.

[26] Horst Sachs. Regular graphs with given girth and restricted circuits. *Journal of the London Mathematical Society*, 1(1):423–429, 1963.

[27] Jakub Smihula. Hamiltonicity of cages. `https://github.com/jakubsmihula/hamiltonicity-of-cages`, 2025. Accessed: 2025-04-12.

[28] Basil Vandegriend. *Finding Hamiltonian cycles: algorithms, graphs and performance.* University of Alberta, 1999.

[29] Ping Zhang and Gary Chartrand. *Introduction to graph theory*, volume 2. Tata McGraw-Hill New York, 2006.

Appendix A: Results of Testing Sufficient Conditions

k	g	Graph Number	Dirac	Ore	Erdős–Hobbs	Jackson
3	3	1	YES	YES	YES	YES
3	4	1	YES	YES	YES	YES
3	5	1	NO	NO	NO	NO
3	6	1	NO	NO	NO	NO
3	7	1	NO	NO	NO	NO
3	8	1	NO	NO	NO	NO
3	9	1	NO	NO	NO	NO
3	9	2	NO	NO	NO	NO
3	9	3	NO	NO	NO	NO
3	9	4	NO	NO	NO	NO
3	9	5	NO	NO	NO	NO
3	9	6	NO	NO	NO	NO
3	9	7	NO	NO	NO	NO
3	9	8	NO	NO	NO	NO
3	9	9	NO	NO	NO	NO
3	9	10	NO	NO	NO	NO
3	9	11	NO	NO	NO	NO
3	9	12	NO	NO	NO	NO
3	9	13	NO	NO	NO	NO
3	9	14	NO	NO	NO	NO
3	9	15	NO	NO	NO	NO
3	9	16	NO	NO	NO	NO
3	9	17	NO	NO	NO	NO
3	9	18	NO	NO	NO	NO
3	10	1	NO	NO	NO	NO
3	10	2	NO	NO	NO	NO
3	10	3	NO	NO	NO	NO

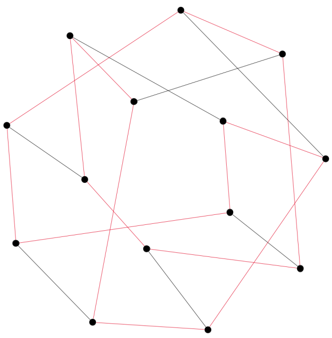
k	g	Graph Number	Dirac	Ore	Erdős–Hobbs	Jackson
3	11	1	NO	NO	NO	NO
3	12	1	NO	NO	NO	NO
4	5	1	NO	NO	NO	NO
4	7	1	NO	NO	NO	NO
5	5	1	NO	NO	NO	NO
5	5	2	NO	NO	NO	NO
5	5	3	NO	NO	NO	NO
5	5	4	NO	NO	NO	NO
6	5	1	NO	NO	NO	NO
7	5	1	NO	NO	NO	NO
7	6	1	NO	NO	NO	NO

Table 5.3: Results of Tested Sufficient Condition in Cages

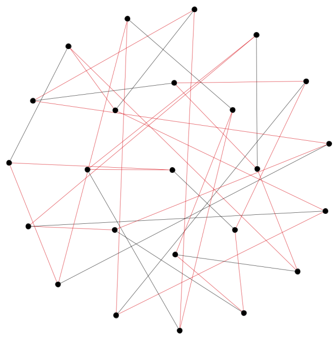
k	g	Graph Number	Dirac	Ore	Erdős–Hobbs	Jackson
3	14	1	NO	NO	NO	NO
3	16	1	NO	NO	NO	NO
3	17	1	NO	NO	NO	NO
3	18	1	NO	NO	NO	NO
3	20	1	NO	NO	NO	NO
3	23	1	NO	NO	NO	NO
3	25	1	NO	NO	NO	NO
3	29	1	NO	NO	NO	NO
3	30	1	NO	NO	NO	NO
4	9	1	NO	NO	NO	NO
4	10	1	NO	NO	NO	NO
5	10	1	NO	NO	NO	NO
7	8	1	NO	NO	NO	NO
10	5	1	NO	NO	NO	NO
12	5	1	NO	NO	NO	NO

Table 5.4: Results of Tested Sufficient Conditions in Record graphs

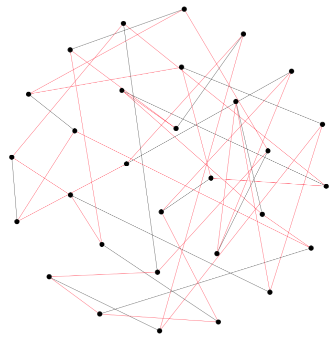
Appendix B: Visual Results of Hamiltonicity Test



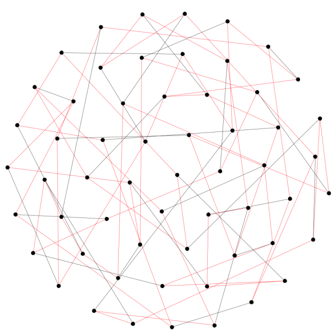
(a) $(3, 6)$-cage



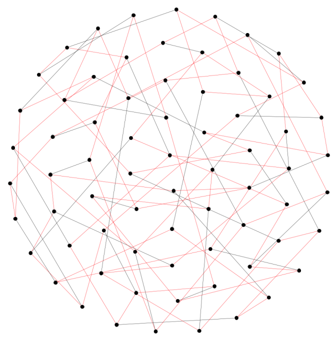
(b) $(3, 7)$-cage



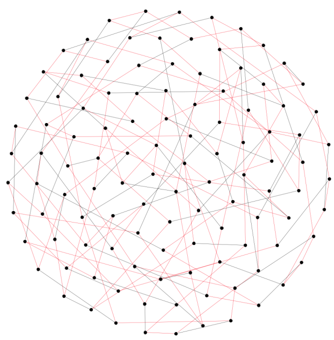
(c) $(3, 8)$-cage



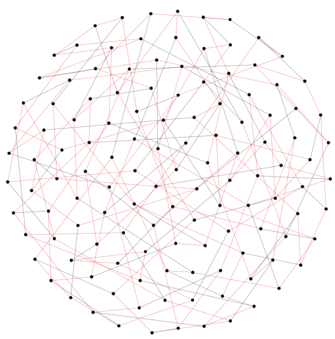
(d) $(3, 9)$-cage



(e) $(3, 10)$-cage

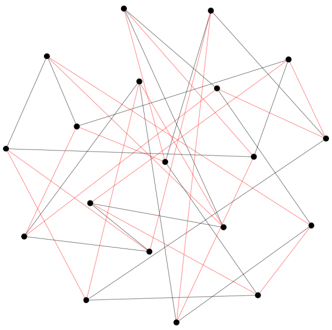


(f) $(3, 11)$-cage

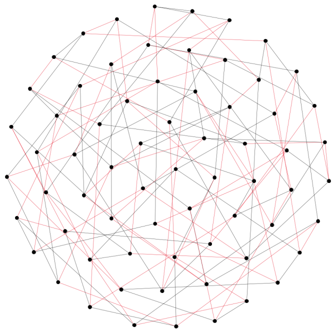


(g) $(3, 12)$-cage

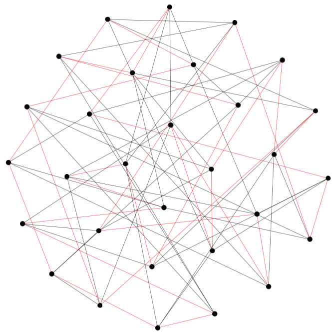
Figure 5.10: Hamiltonian cycle in cages with $k = 3$



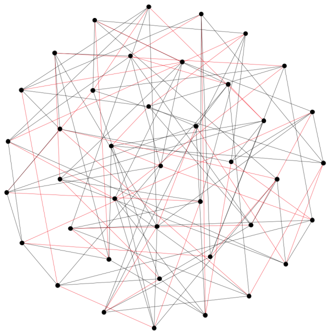
(a) $(4,5)$-cage



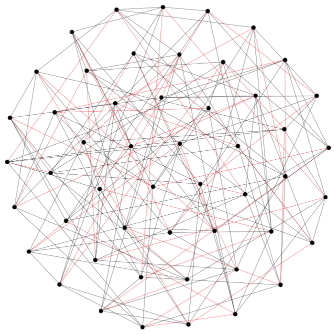
(b) $(4,7)$-cage



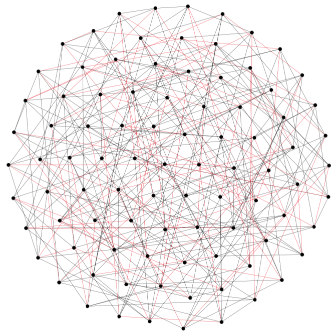
(c) $(5,5)$-cage



(d) $(6,5)$-cage

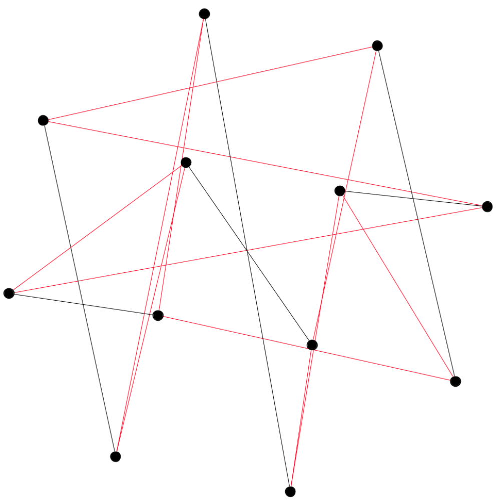


(e) $(7,5)$-cage

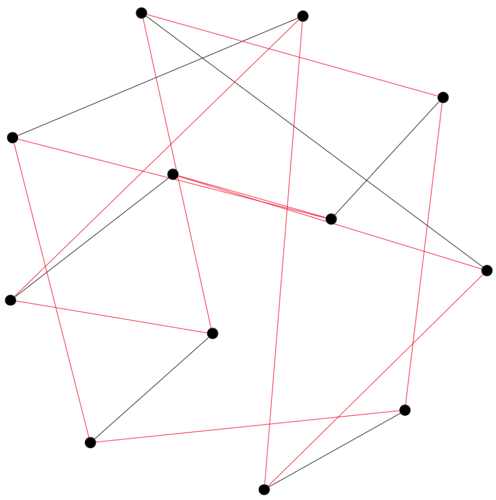


(f) $(7,6)$-cage

Figure 5.11: Hamiltonian cycles in other (k,g)-cages, where $k \in \{4,5,6,7\}$

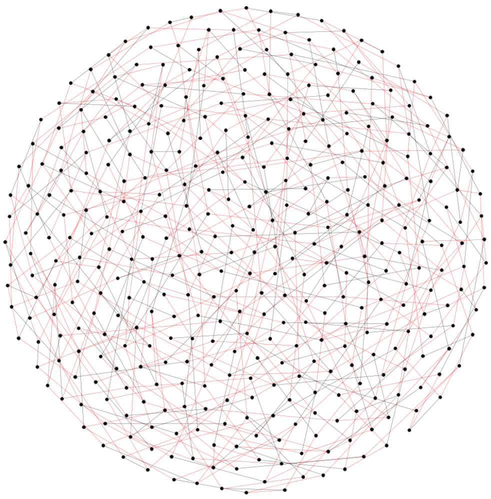


(a) $(3,5)$-graph with excess 2

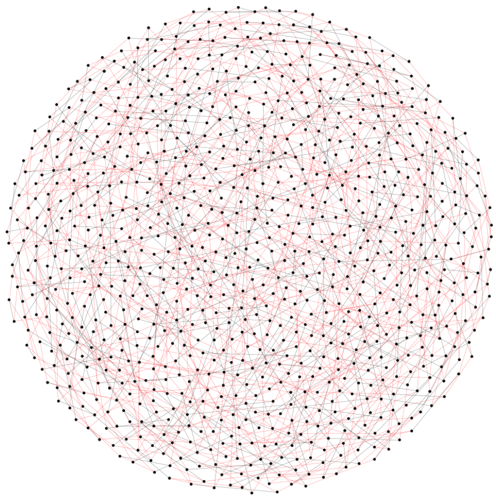


(b) $(3,5)$-graph with excess 2

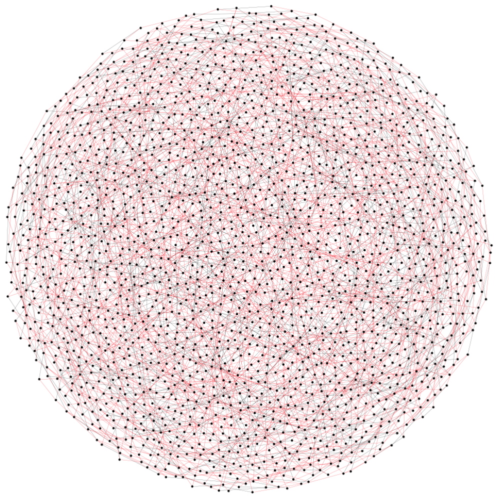
Figure 5.12: Hamiltonian cycles in $(3,5)$-graphs with excess 2



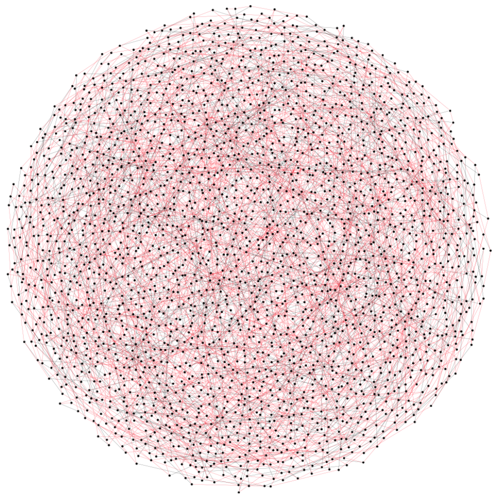
(a) $\mathrm{rec}(3, 14)$



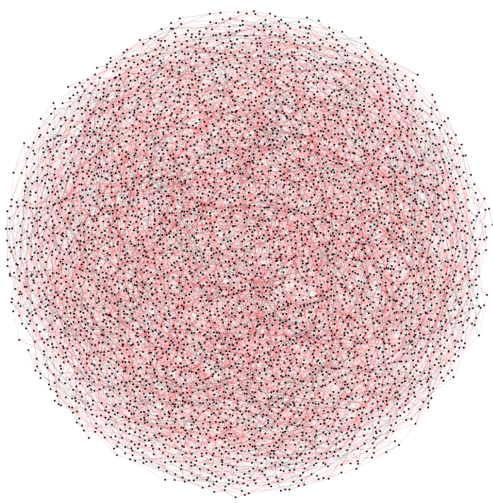
(b) $\mathrm{rec}(3, 16)$



(c) $\mathrm{rec}(3, 17)$

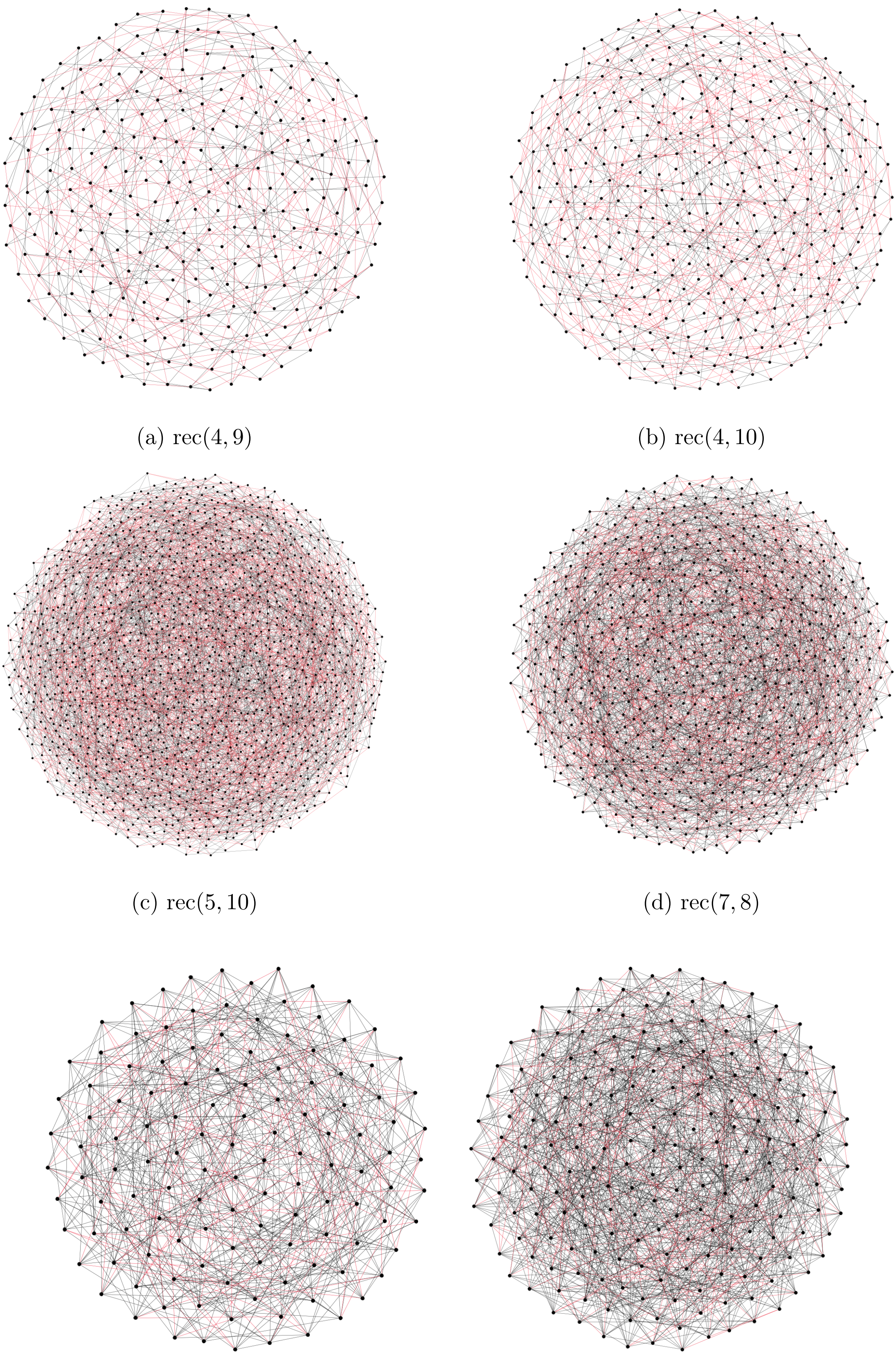


(d) $\mathrm{rec}(3, 18)$



(e) $\mathrm{rec}(3, 20)$

Figure 5.13: Hamiltonian cycles in $\mathrm{rec}(3, g)$ graphs



(a) rec$(4, 9)$

(b) rec$(4, 10)$

(c) rec$(5, 10)$

(d) rec$(7, 8)$

(e) rec$(10, 5)$

(f) rec$(12, 5)$

Figure 5.14: Hamiltonian cycles in other record graphs.