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 Informatics
Field of Study:	Applied Informatics
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: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Contents

List of Figures

List of Tables

Introduction

+

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.

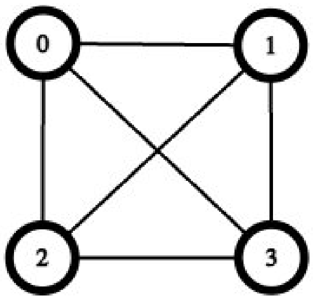


Figure 1.1: Connected undirected simple graph

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) [24] 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.[24]

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.[11]

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 [4]

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.

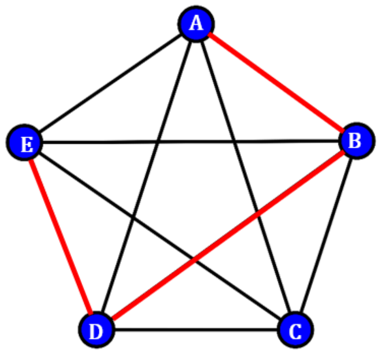
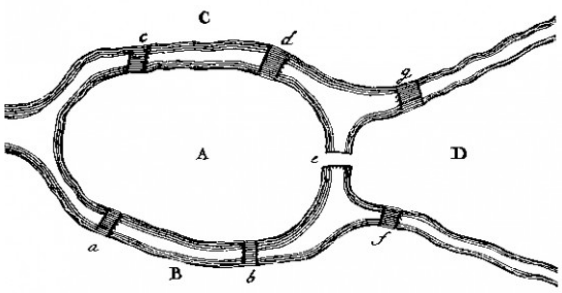


Figure 1.2: K_5 graph

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.



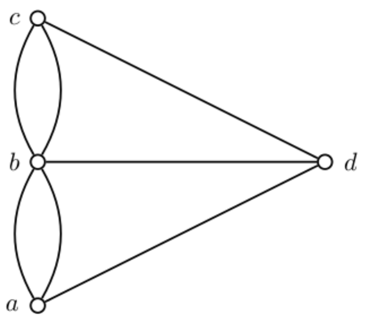


Figure 1.3: The Seven Bridges of Königsberg and corresponding graph

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 Traveling 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.

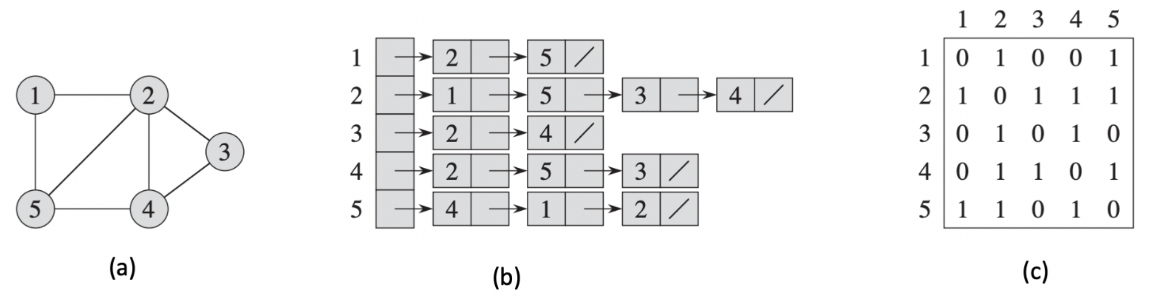
The diameter of the graph is defined as the maximum eccentricity of a vertex that can be found in a graph.

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[6]

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. [6]

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.[10] 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.

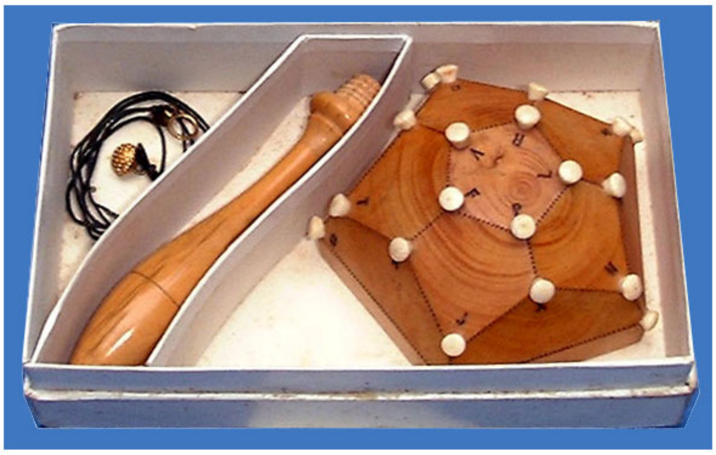


Figure 2.1: Icosian game

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. [12]

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 $P = x_0 x_1 ... x_k$ 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 $x_0 x_{i+1} \in E$ and $x_i x_k \in E$. Then the Hamiltonian cycle is formed as

$$C = x_0 x_{i+1} x_{i+2} \dots x_{k-1} x_k x_i x_{i-1} \dots x_1 x_0$$

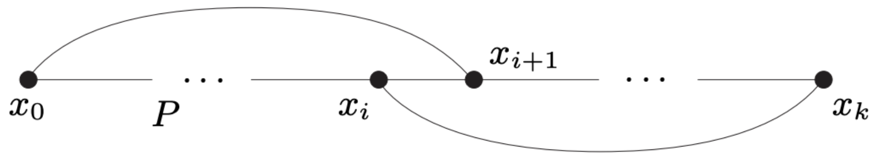


Figure 2.2: Hamilton cycle in a proof of Dirac's Theorem

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 $y x_j \in E$ for some $0 \leq j \leq k$. But then a path $P' = y x_j, x_j + ...$ 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 [8] 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 [8]

2.2.3 Jackson

Jackson in [13] 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.[23]. 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. [23]

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. [16]

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. [6]

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. [6]

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. [6]

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.[6]

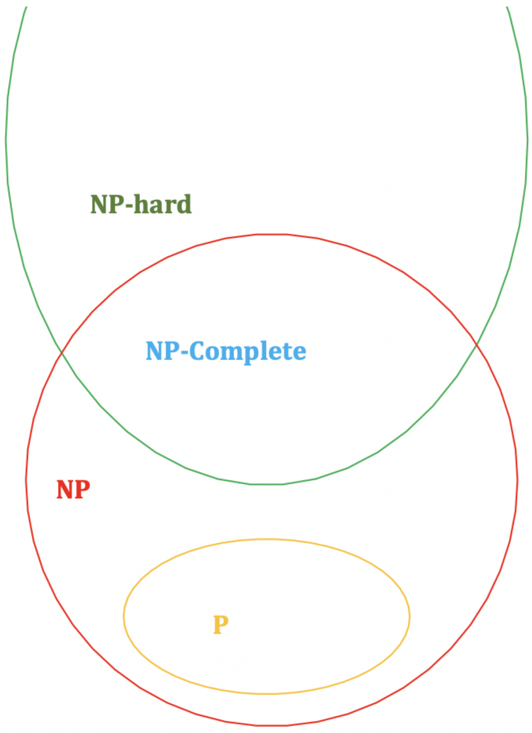


Figure 2.3: Complexity classes where $NP \neq P$

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.[9]

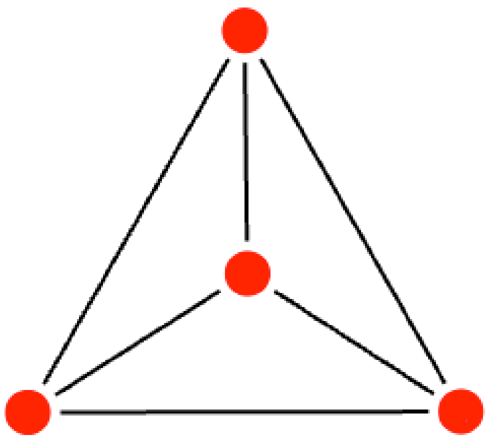


Figure 3.1: (3,3)-Cage

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 $(k, g) - \mathit{cage}$ 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 $g = 6$ [14][9].

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.[17]. The Moore bound is denoted as $M(\Delta, d)$.

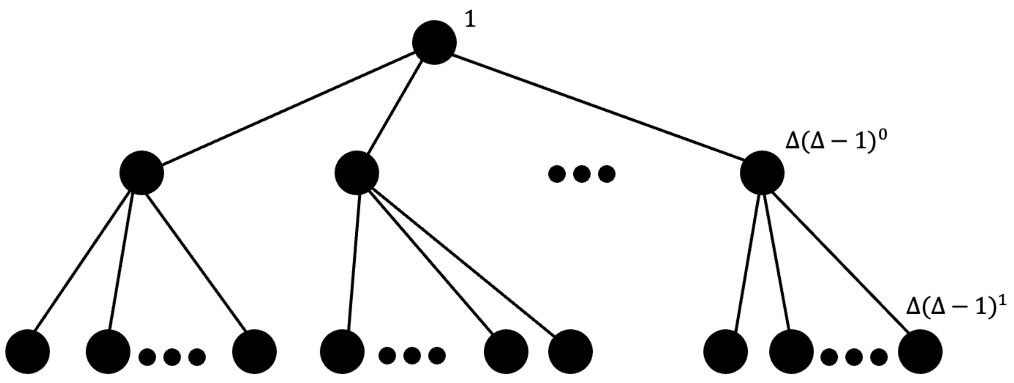


Figure 3.2: The idea behind deriving Moore Bound

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.[17]

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 $\Delta = k$. [9]

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.

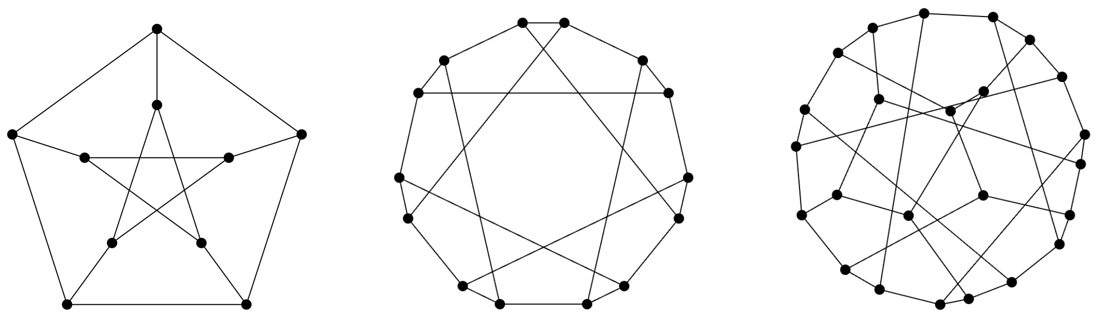
k	g	Name	# of Cages	Order
3	5	Petersen Graph	1	10
3	6	Heawood Graph	1	14
3	7	McGee Graph	1	24
3	8	Tutte-Coxeter Graph	1	30
3	9		18	58
3	10		3	70
3	11	Balaban Graph	1	112
3	12	Benson Graph	1	126
4	5	Robertson Graph	1	19
5	5		4	30
6	5		1	40
7	5	Hoffman-Singleton Graph	1	50
7	6		1	90
4	7		N/A	67

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.[19]



(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 $g \geq 2$.[21]. When order of (k, g)-cage is unknown we denote the smallest known k-regular graph with girth g as Record graph - $rec(k, g)$.

tab-
t viac
j od
chadzaju

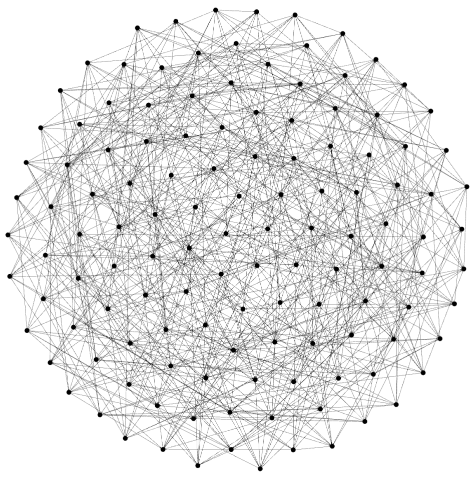
k	g	Order
3	14	384
3	16	960
3	17	2176
3	18	2560
3	20	5376
4	9	275
4	10	384
5	10	1296
7	8	672
10	5	124
12	5	203

Table 3.3: Record graphs for various values of degree k and girth g.

u?

3.5 Construction of regular hamiltonian graphs with given girth

Sachs in [21] 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.


(a) $rec(10, 5)$

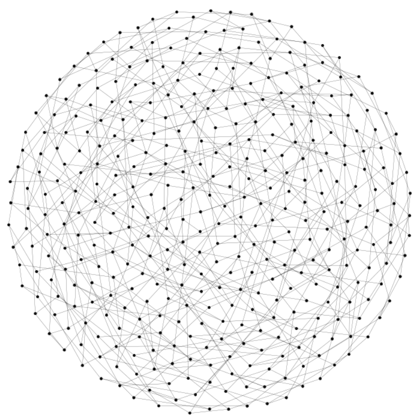

(b) $rec(3, 14)$

Figure 3.4: Record graphs

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 [15], 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.

Algorithm 1 Sachs Construction

```
1: Procedure SachsConstruction($k, g$)
2: if $g = 2$ then
3:     return $G \leftarrow$ multigraph with two vertices and $k$ edges
4: end if
5: if $k = 2$ then
6:     return $G \leftarrow$ cycle graph of length $g$
7: end if
8: $G_\Delta \leftarrow$ SachsConstruction($k-1, g$)
9: $G_\Lambda \leftarrow$ SachsConstruction($|V(G_\Delta)|, g-1$)
10: $G \leftarrow$ ReplaceVerticesWithGraph($G_\Delta, G_\Lambda$)
11: return $G$
```

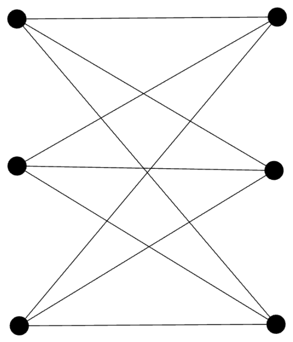
The inductive step reduces the problem by constructing two simpler graphs:

- 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 $g-1$.

Algorithm 2 Replace Vertices With Graph

```
1: **Procedure** ReplaceVerticesWithGraph($(\Delta, \Lambda)$)
2: $F \leftarrow$ empty graph
3: **for** $i = 1$ to $\lambda$ **do**                                  ▷ $\lambda = |V(\Lambda)|$
4:     Create disjoint copy $\Delta_i$ of $\Delta$
5:     Relabel vertices of $\Delta_i$ uniquely
6:     Add $\Delta_i$ to $F$
7: **end for**
8: **for** $i = 1$ to $\lambda$ **do**
9:     Connect a fixed endpoint of $\Delta_i$ to the next $\Delta_{i+1 \pmod{\lambda}}$ along the cycle
10: **end for**
                              ▷ $H(\Lambda)$ - edges of Hamiltonian cycle in $\Lambda$
11: **for** each remaining edge $(v_i, v_j) \in E(\Lambda) \setminus H(\Lambda)$ **do**
12:     Select an unused vertex $u$ from $\Delta_i$
13:     Select an unused vertex $w$ from $\Delta_j$
14:     Add $(u, v)$ to $E(F)$
15: **end for**
16: **return** $F$
```

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


(a) $(3, 4)$-cage

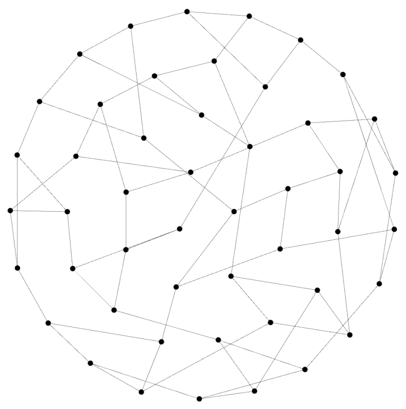

(b) Hamiltonian $(3, 4)$ graph by Sachs

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 [22].

3.6 Open problems

Do motivacie pojde ze preco nas to zaujima.

Do motivacie - v clanku o mixed graphs - pri konstrukcii mixed graphs je konstrukcia ze sa najde hamiltonovka kruznica a sa zorientuju jej grafy. Pri konstrukciach je bvyhodou ked exituje kruznica v grafe.

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 softwares

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 [7]:

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 [22].

4.1.2 Concorde

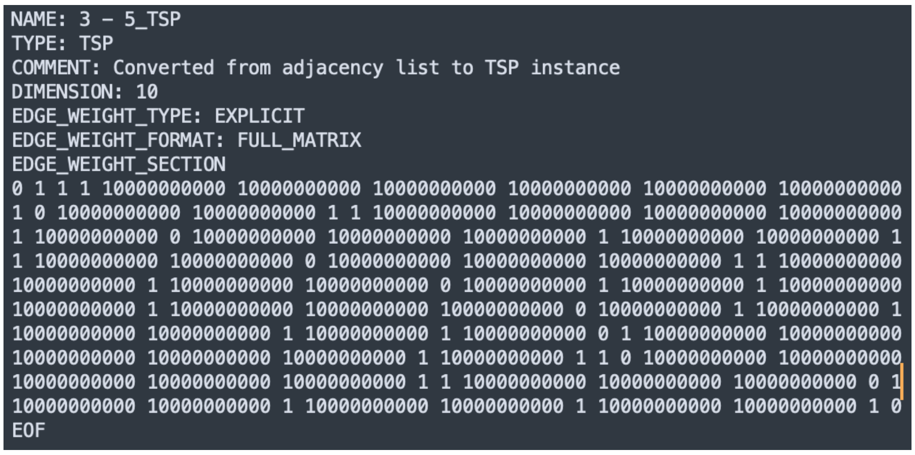
The main software used in our work is Concorde. Concorde is a highly efficient and widely used tool for solving the TSP and several related combinatorial optimization problems. One of its benefits is its ability to compute optimal solutions to TSP instances, making it a standard tool in the field of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

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`. [20]

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. [3] 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. [5] 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: , which would not be feasible on regular computers.

re which
ere
n De-

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.

5.1.1 Cages Datasets

Cages are well-known, relatively small graphs. For this experiment, we used a collection of cage graphs presented in [9]. Their explicit representations are available on the House of Graphs website [18]. 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 [18]

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

Exoo

5.2 Data transformations

5.2.1 Inputted Data format

5.2.2 Reduction of HCP to TSP

5.2.3 TSPLIB format - input for Concorde

5.2.4 Cycle representation - output from Concorde

How are graph represented, in, out of Concorde.

Graph transformation from Hamiltonicity to TSP

5.3 Experimental Procedure

5.3.1 Verifying Sufficient Conditions

5.3.2 Testing Hamiltonicity in Cages

5.3.3 Testing Hamiltonicity in Regular graphs

Mention (3,18) - and problem with it - add image of (3,18) and maybe (3,20).

5.4 Reproducibility and Data Access

Where are located data from experiment

Todo list

Conclusion

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] 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.

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

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

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

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

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

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

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

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

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

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

[14] 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).

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

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

[17] 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.

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

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

[20] 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.

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

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

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

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