Book Drawing Algorithm for Fixed Vertex Order Graph

M.Tech Thesis

by

Dhananjay Suresh Kansule

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025

Book Drawing Algorithm for Fixed Vertex Order Graph

A THESIS

Submitted in partial fulfillment of the

requirements for the award of the degree

of

Master of Technology

by

Dhananjay Suresh Kansule

2302101009

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Book Drawing Algorithm for Fixed Vertex Order Graph** in the partial fulfillment of the requirements for the award of the degree of **Master of Technology** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the period from July 2023 to May 2025 under the supervision of Dr. Ranveer Singh, Indian Institute of Technology Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

19/May/2025
Signature of the Student with Date

(Dhananjay Suresh Kansule)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

19/May/2025

Signature of Thesis Supervisor with Date

(Dr. Ranveer Singh)

Dhananjay Suresh Kansule has successfully given his M.Tech. Oral Examination held on **30/April/2025**.

Signature(s) of Supervisor(s) of M.Tech. thesis

Date: 19/May/2025

Subhra Mazumdar

Signature of Chairman, PG Oral Board

Date: 18.05.2025

Signature of HoD

Date: 18-May-2025

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, **Dr. Ranveer Singh**, Assistant Professor at the Indian Institute of Technology Indore, for his invaluable guidance, continuous support, and encouragement throughout the course of my M.Tech thesis work. His deep insights, constructive feedback, and unwavering patience have been instrumental in shaping my understanding of the subject and bringing this research to fruition. I am immensely thankful for the opportunity to work under his mentorship, which has been both a privilege and a source of inspiration.

I would also like to extend my sincere thanks to my collaborators and co-authors, **Geetanath Gonam** , a dedicated B.Tech student, and **Hareshkumar Jadav**, a passionate Ph.D. scholar, for their active involvement and enthusiastic contribution to this work. Their collaborative spirit, technical expertise, and thoughtful discussions greatly enhanced the quality and direction of our research. Working alongside them has been a highly enriching and rewarding experience.

This thesis, titled *"Book Drawing Algorithm for Fixed Vertex Order Graph"*, is the result of a collaborative journey, and I am truly fortunate to have had the support of such dedicated individuals throughout.

I also take this opportunity to express my heartfelt appreciation to **Professor Suhas Joshi**, Director of IIT Indore, for providing a world-class academic environment and the necessary infrastructure that enabled me to pursue my research effectively. The vibrant research culture and supportive academic atmosphere at IIT Indore played a crucial role in my overall growth as a researcher.

My gratitude also extends to all the faculty members of the Department of Computer Science and Engineering at IIT Indore for their teachings and encouragement, which have deeply enriched my academic experience. I also acknowledge the support staff for their administrative assistance during the course of my M.Tech program.

Dhananjay Suresh Kansule

M.Tech, Department of Computer Science and Engineering

Indian Institute of Technology Indore

Dedicated to My Family

ABSTRACT

Book drawing is a type of graph embedding in which the vertices of a graph G are placed along a straight line called the spine of a book, and the edges are assigned to the pages so that each page contains a subset of the edges. The main goal in book drawing is to minimize the number of pages required while keeping the number of crossings per edge bounded by a non-negative integer b.

In this paper, we present a heuristic algorithm that produces a book drawing for any graph in polynomial time. The algorithm assigns rankings to the edges of the graph, and using these rankings along with the crossing bound as parameters, it generates the book drawing of the graph.

We evaluate our algorithm through experiments on two types of graphs: randomly generated graphs using the Erdős–Rényi model and randomly generated regular Hamiltonian graphs. In addition, we present results on several well-known graphs to show how our algorithm distributes the edges across pages under different crossing bounds.

Contents

List of Figures

List of Tables

NOTATIONS

1. σ : Vertex order

2. $cr(e)$: Number of crossings of edge e

3. $df(e)$: Vertex difference in edge e, that is, $\sigma(v) - \sigma(u)$

Chapter 1

Introduction

1.1 Graph and Graph Drawing

A graph is a fundamental structure in discrete mathematics and theoretical computer science, consisting of a set of vertices and a set of edges, where each edge connects a pair of vertices. Formally, a graph G is defined as an ordered pair (V, E), where V is the set of vertices and $E \subseteq \{\{u, v\} \mid u, v \in V\}$ is the set of edges. A simple graph with 4 vertices and 4 edges is shown in Figure 1.1. Graphs can be either undirected or directed, depending on whether the edges indicate direction. They can also be weighted or unweighted, depending on whether a numerical value (weight) is associated with each edge. Graphs are used to model a wide range of systems and structures such as social networks, communication networks, transportation systems, and biological networks.

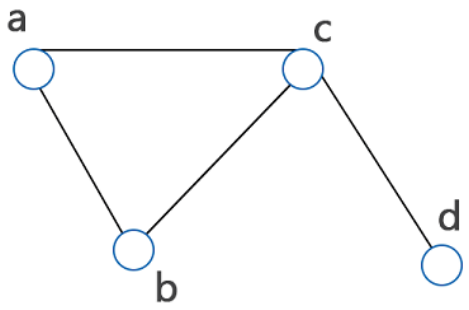


Figure 1.1: Simple Graph

Graph drawing is the field concerned with the geometric representation of graphs. The primary objective in graph drawing is to produce a clear, readable, and informative visual depiction of the graph's structure. This involves assigning coordinates to vertices and drawing edges—usually as straight lines or curves—so that the resulting image conveys the relationships among the vertices effectively. One of the key challenges in graph drawing is minimizing visual complexity, such as reducing the number of edge crossings, edge lengths, or angles between edges. In many applications, especially in theoretical research, constraints may be imposed on the drawing, such as fixing the order of vertices, restricting edges to lie within certain regions, or minimizing the number of pages in a book embedding.

Types of Graph Drawing

1. **Planar Drawing:**

   The goal is to draw the graph on a plane without any edge crossings. This is possible only for *planar graphs*. Such drawings are useful for minimizing visual clutter and improving readability.

2. **Straight-Line Drawing:**

   All edges are represented as straight-line segments. This type of drawing is commonly used for its simplicity and clarity in visualizing relationships between vertices.

3. **Circular Drawing:**

   Vertices are arranged on the circumference of a circle. The focus is often on minimizing edge crossings or total edge length. This type is popular for visualizing cyclic structures or symmetrical graphs.

4. **Layered (Hierarchical) Drawing:**

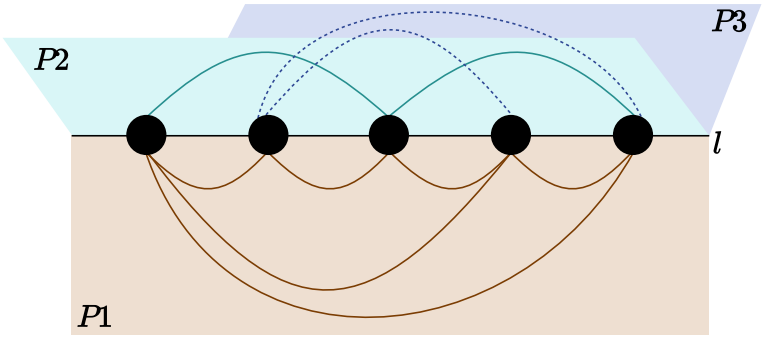


Figure 1.2: Book embedding of K_5

Vertices are placed in multiple layers, usually in a top-down or left-to-right manner. It is especially useful for directed acyclic graphs (DAGs), such as flowcharts, where edge direction matters.

5. **Book Drawing:**

   Vertices are placed along a straight line called the *spine*, and edges are assigned to pages (half-planes bounded by the spine). The objective is typically to minimize the number of pages or crossings per page.

1.2 Book Drawing

A *book drawing* of a graph G is a type of graph drawing where the vertices, with an ordering σ, are placed along a single line called the *spine* of the book, and the edges are assigned to one or more *pages*. A *book embedding* is a book drawing with zero crossings on all pages. A book embedding of the complete graph with 5 vertices, K_5, on 3 pages is shown in Figure 1.2.

The main objective in book drawing is to minimize certain parameters, such as the total number of pages required to draw the graph without any edge crossings on the same page (called the *book thickness* or *page number* of the graph), or to minimize the number of crossings if a fixed number of pages is given. In some variations, a fixed vertex order is given on the spine, and the

challenge is to assign edges to pages in a way that optimizes the drawing. From a complexity perspective, computing the minimum number of pages needed for a book embedding of a graph is known to be **NP**-hard in general.

1.3 Applications of Book Drawing

1. **VLSI Design (Very Large Scale Integration):**

   In designing microchips, wires (edges) connecting components (vertices) should cross as little as possible to reduce signal delay and manufacturing complexity. Book embeddings help in assigning these connections to different layers (pages) while minimizing crossings.

2. **Parallel Processing:**

   Tasks in parallel systems are represented as graphs where edges represent data dependencies. Book embeddings can be used to schedule these tasks across different processors (pages) in a way that minimizes conflicts and ensures efficiency.

3. **Graph Drawing & Visualization:**

   Book embeddings allow large, complex graphs to be visualized more clearly by distributing edges across multiple pages. This helps in applications like software engineering, where understanding class hierarchies or function calls is crucial.

4. **Computational Geometry:**

   Certain geometric problems, like organizing overlapping objects or layouts in layers, can be modeled using book embeddings. These provide a clean separation of elements with minimized overlap.

5. **Database Schemas & Data Organization:**

In database design, especially hierarchical or relational databases, book embeddings help represent entity-relationship models or data flow diagrams in an organized, layered manner.

6. **Bioinformatics:**

   Book drawings are used to model the folding and arrangement of chromosomes or protein structures. This helps researchers understand how sequences are structured spatially with minimal entanglement.

7. **Theoretical Computer Science:**

   Concepts like *page number* and *book thickness* of graphs are actively studied in graph theory. These help in classifying graphs and designing efficient algorithms for storage and retrieval.

8. **Network Routing:**

   In optical or wireless networks, efficient path routing with minimal interference is critical. Book embeddings help in planning routes across different frequencies or channels (pages) to reduce collision and congestion.

1.4 Preliminaries

Given a graph $G = (V, E)$, with vertex set V and edge set E. The vertices are placed on a spine in a fixed order $\sigma = \{v_1, v_2, \ldots, v_n\}$. For an edge (u, v) in the graph (G, σ), we assume $\sigma(u) < \sigma(v)$. In this fixed vertex order, two edges $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ cross each other if both are placed on the same page and their endpoints interleave, specifically if $\sigma(u_1) < \sigma(u_2) < \sigma(v_1) < \sigma(v_2)$ or if $\sigma(u_2) < \sigma(u_1) < \sigma(v_2) < \sigma(v_1)$ as shown in Figure 1.3a. The *contained edges* of an edge $e = (u, v)$ are the set of edges $\{e' = (u', v') : e \neq e'$ and $\sigma(u) \leq \sigma(u') \leq \sigma(v') \leq \sigma(v)\}$, and e is an *outer edge* of this set of edges as shown

in Figure 1.3b. $cr(e)$ denotes the number of crossings of edge e. For edge $e = (u, v)$, $df(e)$ denotes $\sigma(v) - \sigma(u)$.

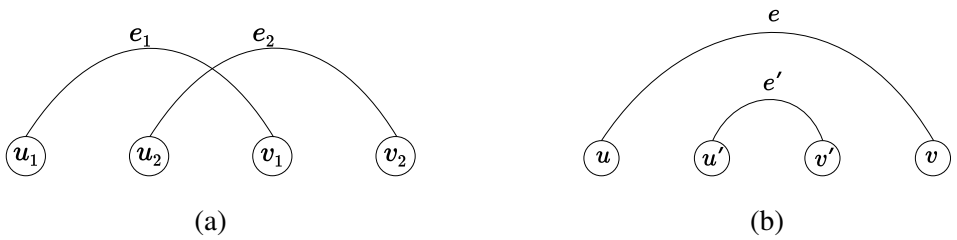


Figure 1.3: (a) Crossing edges; (b) Contained edges

The **book thickness** or **page number** of G is the minimum number of pages in a book embedding of G. Determining the page number is an NP-complete problem.

A **conflict graph** is a graph constructed from graph G where each vertex in the conflict graph represents an edge from G. An edge exists between two vertices in the conflict graph if and only if the corresponding edges in G cross each other. Specifically, each page in the book embedding corresponds to a color in the conflict graph. Finding the page number of a graph G with a fixed vertex order is equivalent to determining the k-colorability of its conflict graph.

Chapter 2

Literature Survey

Book embedding was first introduced by Kainen [14] and was later examined in depth by Bernhart and Kainen [7], who proved that any complete graph K_n can be embedded in $\lceil \frac{n}{2} \rceil$ pages (Theorem 3.4 of [7]). In 1987, Chung et al. [10] introduced a new perspective on book embeddings by examining the concept of **page width**, defined as the maximum number of edges intersected by any line perpendicular to the spine on a single page. They demonstrated that K_n admits a $\lceil \frac{n}{2} \rceil$-page book embedding of width n (Proposition 3.6 of [10]). Later, in 1998, Bilski [9] improved this result by showing that K_n could be embedded in $\lceil \frac{n}{2} \rceil$ pages with reduced width $n-3$, providing an optimal bound on the width without requiring any specific vertex ordering.

Improved upper bounds on the book thickness of graphs are often derived through meta-theorems that leverage classical graph parameters. For instance, it has been shown that a graph with m edges has a book thickness of $O(\sqrt{m})$ [16], and similarly, graphs with genus g have a book thickness bounded by $O(\sqrt{g})$ [17]. Furthermore, Dujmović and Wood demonstrated that graphs with treewidth w have book thickness at most $w + 1$ [11], refining a previous linear bound established by Ganley and Heath [13].

It is also established that all graphs within a minor-closed family possess

bounded book thickness [22]. However, the converse does not always hold true. For example, although 1-planar graphs do not form a minor-closed family [2], they still maintain a bounded book thickness [3, 4]. Recall that a graph is said to be h-planar, for $h \geq 0$, if it can be embedded in the plane such that each edge experiences at most h crossings; see [20, 21] for recent overviews on this topic.

Significantly, the methods discussed in [5, 6] represent the first substantial adaptations of the level-based peeling strategy introduced by Yannakakis [24, 25] to non-planar graph classes. These techniques leverage a key structural characteristic of 3-connected 1-planar graphs—namely, that such graphs can be enhanced and drawn so that each pair of intersecting edges is confined within a degree-4 face of a planar skeleton. This skeleton includes all the vertices and the non-crossing edges of the drawing [23].

A comparable structural property is observed in optimal 2-planar graphs. For these, it is possible to construct drawings where the planar skeleton remains simple and biconnected, with each face having degree 5 and enclosing five crossing edges [18]. Despite these advancements in structural understanding, the book thickness of these graphs has not yet been thoroughly investigated. The best known upper bound remains $O(\log n)$, inherited from general results on h-planar graphs [19].

In 2020, Bhore et al. [8] investigated fixed-parameter tractable algorithms for book drawing problems, parameterized by vertex cover and path width. They developed algorithms with running times of $2^{O((\mathrm{vc})^3)}n$ and $(\mathrm{pw})^{(\mathrm{pw})^2}n$, respectively, where vc denotes the vertex cover number and pw denotes the path width.

In 2020, Liu et al. [15] improved upon this work by addressing the problem of fixed-order book drawing with a bounded number of crossings per edge. They developed algorithms to determine whether there exists a k-page book drawing

with at most b crossings per edge, with running times of $(b+2)^{O((\text{vc})^3)}n$ and $(b+2)^{O((\text{pw})^2)}n$.

In 2024, Agrawal et al. [1] addressed the problem of edge deletion to achieve a k-page graph with at most b crossings per edge. Additionally, they solved the problem of e-edge deletion to obtain a 1-page graph with at most b crossings per edge in $2^{O(d\sqrt{k}\log(d+k))} \cdot n^{O(1)}$. Furthermore, they also addressed the e-edge deletion k-page problem with a running time of $4^m \cdot n^{O(1)}$. They developed an algorithm for finding page number with a running time of $2^m \cdot n^{O(1)}$, where m is the number of edges in the graph. But this algorithm is specifically for book embedding, and this algorithm takes exponential time.

Our contribution

We propose a book drawing algorithm for a graph $G = \{V, E\}$. First, in chapter 3 we design book drawing algorithm for any graph. In section 3.1, we create a book embedding structure of graph G on $\lceil n/2 \rceil$ pages based on the embedding structure for the complete graph K_n, including only those edges present in G. Next, in section 3.2, we assign rankings to the edges. Finally, in section 3.3, we develop a page assignment algorithm that uses the ranking of edges to distribute edges across different pages in the book drawing. In chapter 4, we present the complexity analysis of the algorithm. In Section 5, we provide the book drawing of some graphs. In chapter 6, we provide experimental results of on any random graph generated using Erdős–Rényi model and experimental results on any random regular hamiltonian graph.

Chapter 3

Book Drawing of Graphs with Crossing Bound per Edge

In this chapter, we present an algorithm for the book drawing of a graph. The process begins with constructing a book embedding structure for the complete graph on $\lceil n/2 \rceil$ pages, followed by assigning rankings to its edges. Finally, we introduce a page assignment algorithm that distributes the edges of the graph G across the different pages in the book drawing.

3.1 Book Embedding of Complete Graph

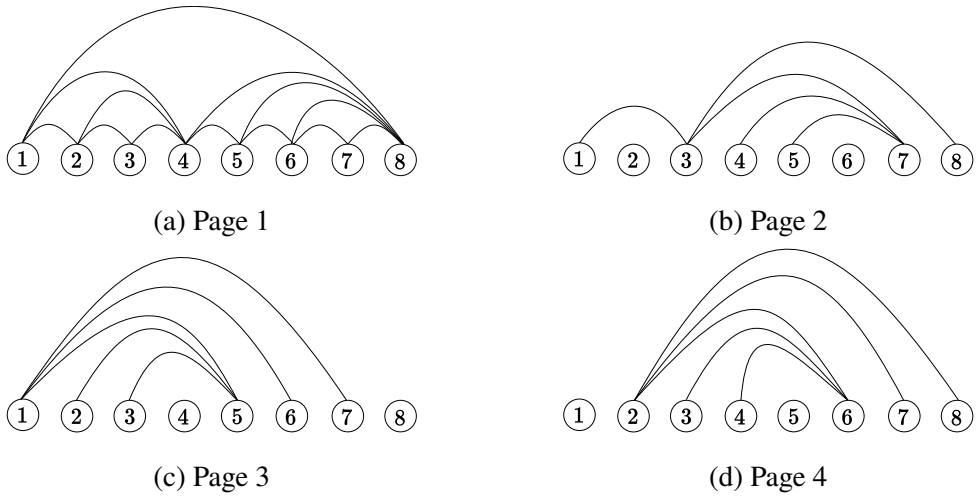
In this section, to embed a complete graph in $\lceil \frac{n}{2} \rceil$ pages, we propose an algorithm that assigns edges to pages using the concept of contained edges to avoid crossings. The contained edges of an edge e will never cross e itself. Based on this fact, if we can add more contained edges so that no crossings occur, we can embed the graph using fewer pages.

The main idea behind this algorithm is that in any complete graph, there would be a set of $\lceil \frac{n}{2} \rceil$ edges $E_m = \{(1, \frac{n}{2} + 1), (2, \frac{n}{2} + 2), (3, \frac{n}{2} + 3), \ldots, (\frac{n}{2}, n)\}$. These edges have the property that each edge from this set crosses all other edges of the set. Thus, we can not place any two edges of this set on the same page

in the book embedding of the complete graph. Therefore, we need at least $\lceil \frac{n}{2} \rceil$ pages to place all these edges on different pages, such that there is no crossing between them. After placing all the E_m edges in $\lceil \frac{n}{2} \rceil$ pages, the remaining edges could be embedded as contained and outer edges of the edges of E_m without any crossing. This idea leads us to embed the K_n in $\lceil \frac{n}{2} \rceil$ pages.

Algorithm 3.1 Page assignment algorithm for K_n

Assign the edge set $\{(j, j+1) \mid 1 \leq j \leq n-1\} \cup \{(1, n)\}$ on page 1
for each page i from 1 to $\lfloor \frac{n}{2} \rfloor - 2$ **do**
    Assign the edges $(1, \lfloor \frac{n}{2} \rfloor - i + 1)$ and $(\lfloor \frac{n}{2} \rfloor - i + 1, n - i + 1)$
    for $j = \lfloor \frac{n}{2} \rfloor - i + 2$ to $n - i - 1$ **do**
        Assign the edge $(j, n - i + 1)$
    end for
    for $j = 2$ to $\lfloor \frac{n}{2} \rfloor - i - 1$ **do**
        Assign the edge $(j, \lfloor \frac{n}{2} \rfloor - i + 1)$
    end for
    for $j = n - i + 2$ to n **do**
        Assign the edge $(\lfloor \frac{n}{2} \rfloor - i + 1, j)$
    end for
end for
Additional Pages:
Page $\lfloor \frac{n}{2} \rfloor - 1$**:**
Assign edges $(1, j)$ for $j = \left\lfloor \frac{(n+1)}{2} \right\rfloor + 1$ to $n - 1$
Assign edges $(j, \left\lfloor \frac{(n+1)}{2} \right\rfloor + 1)$ for $j = 2$ to $\left\lfloor \frac{(n+1)}{2} \right\rfloor - 1$
Page $\lfloor \frac{n}{2} \rfloor$**:**
Assign edges $(2, j)$ for $j = \left\lfloor \frac{(n+1)}{2} \right\rfloor + 2$ to n
Assign edges $(j, \left\lfloor \frac{(n+1)}{2} \right\rfloor + 2)$ for $j = 3$ to $\left\lfloor \frac{(n+1)}{2} \right\rfloor$
if n is odd **then**
    Assign edges $(j, \lfloor \frac{n}{2} \rfloor + 1)$ for $j = 1$ to $\lfloor \frac{n}{2} \rfloor - 1$
end if



(a) Page 1 (b) Page 2

(c) Page 3 (d) Page 4

Figure 3.1: Book embedding of K_8 in 4 pages

Following Algorithm 3.1, K_8 is embedded in 4 pages, as illustrated in Figure 3.1. The algorithm starts by assigning the edges $(1, 2), (2, 3), \ldots, (n-1, n)$, and $(1, n)$, by this we are assigning n edges on the first page (Line 1). Then, for each page i from 1 to $\lfloor \frac{n}{2} \rfloor - 2$, the algorithm assigns two edges, one between vertices 1 and $\lfloor \frac{n}{2} \rfloor - i + 1$, and other between $\lfloor \frac{n}{2} \rfloor - i + 1$ and $n - i + 1$ (Lines 2-3). Next, for each vertex j from $\lfloor \frac{n}{2} \rfloor - i + 2$ to $n - i - 1$, an edge is assigned between j and $n - i + 1$, which adds $\lceil \frac{n}{2} \rceil - 2$ edges to each page (Lines 4-6). The algorithm then assigns edges for j from 2 to $\lfloor \frac{n}{2} \rfloor - i - 1$ and this assigns an edge between vertices j and $\lfloor \frac{n}{2} \rfloor - i + 1$, which adds $\lfloor \frac{n}{2} \rfloor - i - 2$ edges to each page (Lines 7-9). Finally, for j from $n - i + 2$ to n, edges are assigned between vertices $\lfloor \frac{n}{2} \rfloor - i + 1$ and j and this adds $i - 1$ edges on each page (Lines 10-12). By this (lines 1-13) the Algorithm adds exactly $2n - 3$ edges on page 1 and $n - 3$ edges on each page from 2 to $\lfloor \frac{n}{2} \rfloor - 2$. After this (Line 15-20) the Algorithm adds exactly $n - 3$ edges on each of the two pages $\lfloor \frac{n}{2} \rfloor - 1$ and $\lfloor \frac{n}{2} \rfloor$. Now, if n is odd then the Algorithm adds an extra page, which contains exactly $\lfloor \frac{n}{2} \rfloor - 1$ edges.

The Algorithm 3.1 organizes the edges of K_n across the pages, ensuring that crossings for all the edges are zero and minimizing the number of pages used. According to the Algorithm 3.1, we assign each edge of K_n to different pages.

Therefore, it takes $O(n^2)$ time.

Lemma 1. *The Algorithm 3.1 performs books embedding of K_n in $\lceil \frac{n}{2} \rceil$ pages.*

Proof. According to Algorithm 3.1, the first page contains $2n - 3$ edges and pages 2 to $\lfloor \frac{n}{2} \rfloor$ contain $n - 3$ edges each. Therefore, if n is even, then the total number of edges from page 1 to $\lfloor \frac{n}{2} \rfloor$ is

$$(2n - 3) + \left(\left\lfloor \frac{n}{2} \right\rfloor - 1 \right) (n - 3) = \frac{n(n-1)}{2},$$

indicating that all the edges of K_n are assigned across exactly $\frac{n}{2}$ pages. For n being odd, the algorithm uses one extra page containing $\lfloor \frac{n}{2} \rfloor - 1$ edges. Thus, the total number of edges is

$$(2n - 3) + \left(\left\lfloor \frac{n}{2} \right\rfloor - 1 \right) (n - 3) + \left(\left\lfloor \frac{n}{2} \right\rfloor - 1 \right) = \frac{n(n-1)}{2}.$$

Therefore, Algorithm 3.1 embeds any complete graph K_n in exactly $\lceil \frac{n}{2} \rceil$ pages.

□

Consider a graph $G = (V, E)$, with n vertices and m edges. The graph G is essentially a subgraph of the complete graph K_n with the same number of vertices. Since the complete graph K_n can be embedded in $\lceil \frac{n}{2} \rceil$ pages, it implies that any graph with n vertices can potentially be drawn using fewer than $\lceil \frac{n}{2} \rceil$ pages. To draw the graph G in fewer pages, we can rank the edges based on the page assignment of edges of G by Algorithm 3.1. The driving force behind the idea is to rank the edges of the graph G based on the book embedding pattern of the complete graph. The goal is to place more contained edges, as they do not cross each other on the same page. Pages with more contained edges are assigned higher rankings, and the edges on such pages are given a higher ranking.

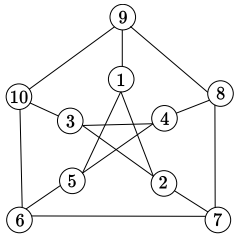
3.2 Edge Ranking Algorithm

In this section, we design an algorithm that takes a graph G with a fixed vertex order σ as input and outputs the ranking of the edges of G. To design such an algorithm, we utilize Algorithm 3.1. We use this ranking as an additional parameter along with the crossing number of edges.

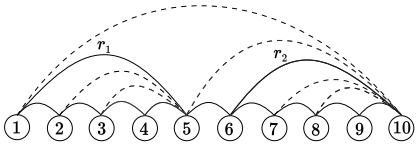
Edges of the graph G with n vertices can be placed on the $\lceil \frac{n}{2} \rceil$ pages according to Section 3.1. Then we assign rankings to the pages first. A page containing more edges than others receives a higher ranking. If multiple pages have the same number of edges, we assign rankings based on the indices of the pages, where a page with a lower index receives a higher ranking. Within each page, rank the edges according to the decreasing order of the difference between the indices of the end vertices. Note that the edges between vertices i and $i + 1$ (where $i = 1$ to $n - 1$), as well as the edge between vertices 1 and n, should not be included when considering the total number of edges on Page 1. Notice that all the edges receive different rankings as shown in Figure 3.2.

Algorithm 3.2 Edge ranking algorithm

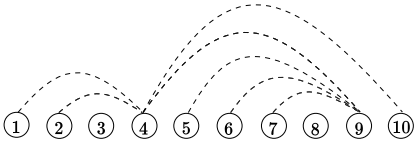
1: **Input:** Graph G with n vertices, vertex order σ
2: **Output:** Ranking of the edges
3: **Initialization:** Place the edges of graph G on $\lceil \frac{n}{2} \rceil$ pages using Section 3.1.
4: Sort the pages based on the total number of edges of G, excluding the edges of type $(i, i + 1)$ for $i = 1$ to $n - 1$, and the edge $(1, n)$.
5: **for** each page $p = 1$ to $\lceil \frac{n}{2} \rceil$ **do** ▷ On sorted pages
6: Rank the edges on p in decreasing order based on the difference between the indices of the end vertices of each edge.
7: **end for**



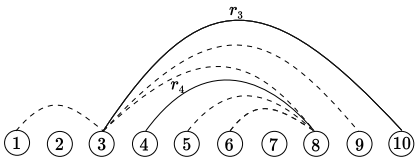
Petersen graph



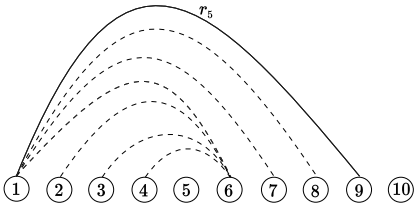
(a) Page 1



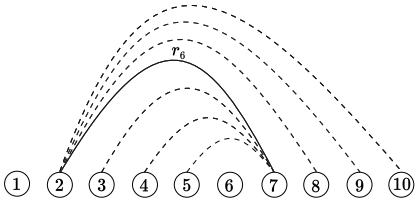
(b) Page 2



(c) Page 3



(d) Page 4



(e) Page 5

Figure 3.2: The Petersen graph is a subgraph of K_{10}. We embed the Petersen graph across 5 pages according to the embedding of K_{10} for any given fixed vertex order. We then count the edges on each page, excluding the edges $(1, 2)$, $(2, 3)$, $\ldots$ $(8, 9)$, $(9, 10)$, and $(1, 10)$. Page 1 and Page 3 has highest number of edges, by indexing of page we assign rank r_1 to edge $(1, 5)$ and r_2 to edge $(6, 10)$ and so on, with $r_1 > r_2 > r_3 > \ldots$. We assign rank to each edge accordingly.

3.3 Page Assignment Algorithm

In this section, we present a page assignment algorithm that minimizes the number of pages in a book drawing. Initially, after ranking all the edges with distinct ranks, all edges of the graph are placed on Page 1, as shown in Figure 3.3. The algorithm begins by computing the number of crossings for each edge. Take edge e with the highest crossing on Page 1, if more than one edge highest crossing on Page 1 then choose the edge with the lowest rank among them.

Initially, the edge e is removed from Page 1 if its crossings exceed b otherwise,

stop. Subsequently, the crossing counts for the remaining edges on Page 1 are updated accordingly. For each removed edge, the algorithm attempts to find a suitable page where the edge can be assigned without exceeding the crossing bound b for any edge. If no such page is found, a new page is allocated. To find a suitable page for e, we shift it from the current page to the next page. The algorithm then checks whether this edge can be safely added without exceeding the crossing bound b for e as well as all edges on that page.

If assigning an edge e to a particular page results in $cr(e) > b$, then check the rankings of all set of edges E' that cross with e. If $cr(e) \leq b$, examine the rankings of all set of edges E' whose crossing bounds exceed b. If any edge $e' \in E'$ has a higher ranking than e, shift e to the next page. If no such edge has a higher ranking, recursively shift edge $e' \in E'$ that exceeds the crossing bound b to the next available page. Among the edges in E', first choose the edge with the highest crossing count; if multiple edges have the highest crossing count, select the one with the lowest ranking, and repeat this process for all edges $e' \in E'$ until $cr(e')$ and $cr(e)$ exceed b. This recursive procedure continues until no edge on any page exceeds the crossing bound b.

- **getMaximumCrossingLeastRankEdge(Edge set E)**: It returns the edge e with the maximum number of crossing in edge set E. If more than one edge has the maximum number of crossings then choose the edge with the lowest rank among them.

addEdgeToPage(edge e, Page i)

1: **while** e is not assigned to Page i **do**
2: **if** e can be assigned to Page i without any edge in Page i exceeding the crossing bound b **then**
3: Assign e to Page i
4: **break**
5: **else**
6: **if** $cr(e) > b$ **then**
7: E' is the set of edges that cross with e
8: **else**
9: E' is the set of edges that cross with e and have $cr(e') > b$ where $e' \in E'$
10: **end if**
11: **if** rank(e) ¡ rank(e') **then** ▷ Any $e' \in E'$ has higher rank than e
12: $i \leftarrow i + 1$
13: **else**
14: Assign e to Page i
15: **for** all edges in E' **do**
16: $e' \leftarrow$ getMaximumCrossingLeastRankEdge(E')
17: **if** $cr(e') \leq b$ and $cr(e) \leq b$ **then**
18: **break**
19: **end if**
20: addEdgeToPage(e', $i + 1$)
21: $E' \leftarrow E' \setminus \{e'\}$
22: **end for**
23: **break**
24: **end if**
25: **end if**
26: **end while**

Algorithm 3.3 Page assignment algorithm

1: **Input:** Graph $G = (V, E)$, edge ranking given by Algorithm 3.2 and crossing bound b
2: **Output:** Book drawing of G with at most b crossings per edge.
3: **Initialization:** Assign all edges E on Page 1
4: Compute the number of crossing for each edge
5: $e \leftarrow$ getMaximumCrossingLeastRankEdge(E)
6: **while** $cr(e) > b$ **do**
7: $i = 2$ ▷ Page i
8: addEdgeToPage(e,i)
9: $E = E \setminus e$
10: $e \leftarrow$ getMaximumCrossingLeastRankEdge(E)
11: **end while**

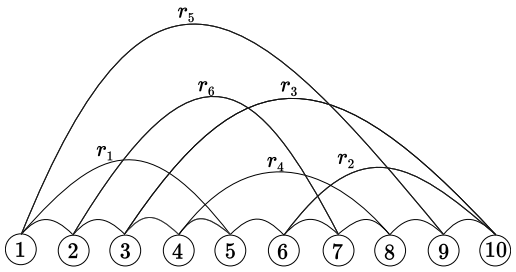


Figure 3.3: Assign all edges of Petersen graph on the first page

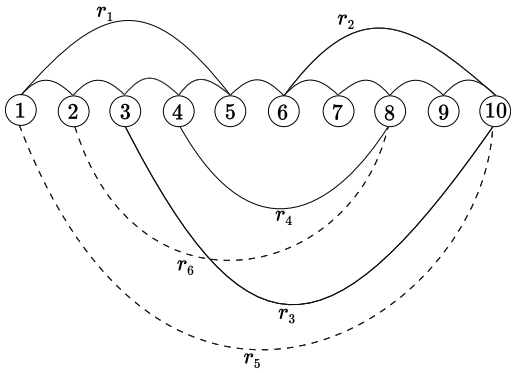


Figure 3.4: The Petersen graph is drawn in 3 pages using the Algorithm 3.3. Page 1 is assigned edges with rank r_1, r_2 and edges $(i, i+1)$ where $i = 1, 2, \ldots, 9$. Page 2 is assigned the edges with rank r_3, r_4 and Page 3 is assigned the edges with rank r_5, r_6

Now, we evaluate the upper bound on the number of pages that Algorithm 3.3 can take for the book drawing of any graph. In doing so, we prove that there is no cyclic dependency in Algorithm 3.3.

Theorem 1. *The Algorithm 3.3 takes at most* $\lceil \frac{n}{2} \rceil + 1$ *pages for the book embedding of any graph G and the Algorithm 3.3 do not contain any cyclic dependency.*

Proof. Let us define E_{p_i}, where $i = 1, 2, 3, \ldots, \lceil \frac{n}{2} \rceil$, as the set of edges on page i of the book embedding of the graph G according to the initialisation step of the Algorithm 3.2. Notice that here indices of pages $i = 1, 2, 3, \ldots, \lceil \frac{n}{2} \rceil$ are assigned after sorting the pages based on the number of edges on them by the Algorithm 3.2. Note that the edges of an E_{p_i} do not cross with each other.

To prove the theorem, we first claim that, the Algorithm 3.3 requires at most two pages to draw E_{p_1} for book embedding of G. Let us assume, we need three pages to draw the edges of E_{p_1} and the pages be P_1, P_1', P_1'' in order according to Algorithm 3.3. Now, Let there be an edge $e \in E_{p_1}$ and it is in P_1''. Note that this is possible only if there is an edge q which can cross with edge e in P_1' (if assigned) and has a higher rank than edge e. Now $q \notin E_{p_1}$ because edges of E_{p_1} can not cross with edge e. So, edge $q \in E_{p_i}$, where $i \in \{2, 3, ..., \lceil \frac{n}{2} \rceil\}$. But all the edges of E_{p_i}, where $i = 2, 3, ..., \lceil \frac{n}{2} \rceil$ have lesser rank than e. Thus by Algorithm 3.3, edge e can not be on P_1''. So edges of E_{P_1} require at most two pages in book embedding. Let us denote the pages required for the edges of E_{P_1} as $\mathcal{P}(E_{P_1})$.

Our second claim is that in the book embedding of G, the Algorithm 3.3 requires $\mathcal{P}(E_{P_1}) \cup \{P_2, P_3, \ldots, P_i\}$ pages to embed the edges of an E_{p_i} where $i \in \{2, 3, ..., \lceil \frac{n}{2} \rceil\}$. We prove this by mathematical induction.

Base Case: According to second claim, edges of E_{P_2} requires $\mathcal{P}(E_{P_1}) \cup P_2$ pages. To prove this, assume E_{P_2} will require one more page say P_2'. Let there be an edge $e_2 \in E_{P_2}$ and it is in P_2'. Note that this is possible only if there

is an edge q_2 that can cross with e_2 in Page P_2 (if assigned) and has a higher rank than edge e_2. Now $q_2 \notin E_{P_2}$ because edges of E_{P_2} can not cross with e_2. Also, $q_2 \notin E_{P_1}$ because edges of E_{P_1} can not be in Page P_2 by the first claim. So, edge $q_2 \in E_{p_j}$, where $j \in \{3, 4, \ldots, \lceil \frac{n}{2} \rceil\}$. But all the edges of E_{p_j}, where $j = 3, 4, \ldots, \lceil \frac{n}{2} \rceil$ have lesser rank than e_2, so this is a contradiction. Hence edges of E_{P_2} can embed in pages $\mathcal{P}(E_{P_1}) \cup P_2$.

Inductive hypothesis: Assume that, the Algorithm 3.3 requires $\mathcal{P}(E_{P_1}) \cup \{P_2, P_3, \ldots, P_k\}$ pages to embed the edges of E_{p_k}.

Inductive step: we have to prove that, the Algorithm 3.3 requires $\mathcal{P}(E_{P_1}) \cup \{P_2, P_3, \ldots, P_{k+1}\}$ pages to embed the edges of $E_{p_{k+1}}$.

Let us assume that one extra page P_{k+2} is required. Let there is an edge $e_3 \in E_{P_{k+1}}$ and it is in P_{k+2} only if there is an edge q_3 that can cross with e_3 in Page P_{k+1} (if assigned) and has a higher rank than edge e_3. Now $q_3 \notin E_{P_{k+1}}$ because edges of $E_{P_{k+1}}$ can not cross with e_3. Also $q_3 \notin E_{P_t}$, where $t \in \{2, 3, \ldots, k\}$ because edges of E_{P_t} can not in Page P_{k+1} . So, edge $q_3 \in E_{p_j}$, where $j \in \{k+2, \ldots, \lceil \frac{n}{2} \rceil\}$. But all the edges of E_{p_j}, where $j = k+2, \ldots, \lceil \frac{n}{2} \rceil$ have lesser rank than e_3, which is a contradiction. Hence edges of $E_{P_{k+1}}$ can embed in Pages $\mathcal{P}(E_{P_1}) \cup \{P_2, P_3, \ldots, P_{k+1}\}$. Therefore, Algorithm 3.3 requires $\mathcal{P}(E_{P_1}) \cup \{P_2, P_3, \ldots, P_i\}$ pages to embed edges of all E_{P_i}, where $i = 2, 3, ..., \lceil \frac{n}{2} \rceil$.

By combining both the claims, the Algorithm 3.3 takes at most $\lceil \frac{n}{2} \rceil + 1$ pages for the book embedding of any graph G. This implies that there is no cyclic dependency in the Algorithm 3.3 as in the case of the cyclic dependency the upper bound can go up to m pages, where m is the number of edges of the graph. □

Chapter 4

Complexity Analysis

In this section, we analyze the overall time complexity of the algorithm proposed for computing the book drawing of a graph. The analysis is broken down into the three core components of the algorithm:

1. **Book Embedding of the Complete Graph:**

    - In this step, all edges of the input graph are embedded across exactly $\lceil \frac{n}{2} \rceil$ pages.
    - 3.1 put all the one by one on $\lceil \frac{n}{2} \rceil$ pages, which takes linear time with respect to a number of edges, i.e., $O(m)$.

2. **Edge Ranking Algorithm:**

    - Edges are initially arranged according to the book embedding of the complete graph K_n.
    - Pages are sorted in descending order based on their edge count. Since there are $\lceil \frac{n}{2} \rceil$ pages, this sorting step requires $O(nlogn)$ time.
    - Within each page, edges are ranked according to the difference in their endpoint vertex indices. As a single page contains at most $n-3$ edges, ranking one page requires $O(n)$ time.

- Therefore, considering all pages, the total time complexity of the edge ranking process is $O(n^2)$.

3. **Page Assignment Algorithm:**

   - For a graph with m edges, counting the number of crossings for an edge initially requires comparing it with every other edge. Therefore, determining the total number of edge crossings requires $O(m^2)$ time.
   - To retrieve the edge e with the maximum number of crossings and the least rank, we use a heap. Therefore, calling `getMaximumCrossingLeastRankEdge()` takes $O(\log m)$ time.
   - Assigning edge e to a different page involves checking it against existing edges, which costs $O(m)$ time.
   - Consequently, handling all edges through this process yields an overall time complexity of:

$$O(m(\log m + m)) = O(m^2).$$

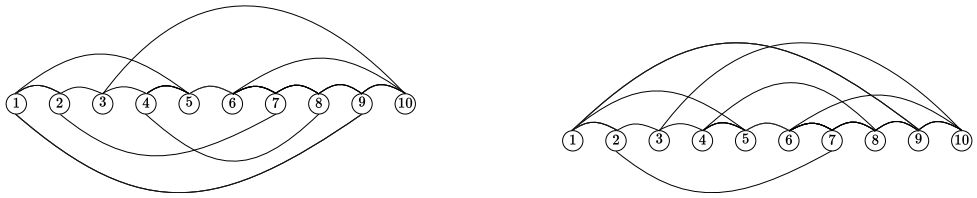
Chapter 5

Example of Book Drawing of Graph

We give some examples to describe the book drawings of certain types of graphs with a crossing bound per edge $b = 0$, 1, and 2. For these graphs, the algorithm gives the book drawing with the optimal number of pages.

1. Petersen graph

Petersen graph book embedding can be done in 3 pages as given in Figure 3.4. Petersen graph book drawings with $b = 1$ and $b = 2$ are shown in Figure 5.1a and Figure 5.1b, respectively.



(a) (b)

Figure 5.1: Petersen graph book drawings in 2 pages with different crossing bounds: (a) when the crossing bound $b = 1$; (b) when the crossing bound $b = 2$.

2. Dürer graph

Dürer graph is a 3-regular generalized Petersen graph with 12 vertices and 18 edges. Dürer graph can be embedded in 3 pages. Book drawing of Dürer graph with $b = 0$, $b = 1$, and $b = 2$ are shown in Figure 5.2a, Figure 5.2b and

Figure 5.2c, respectively.

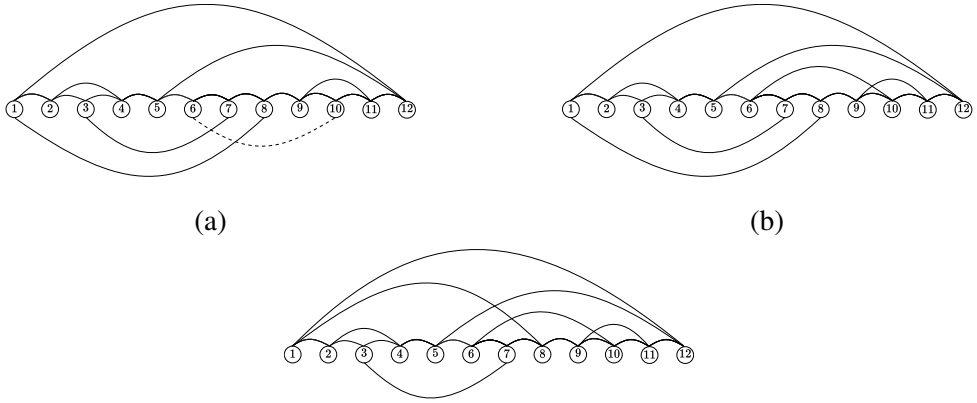


Figure 5.2: Dürer graph book drawings with different page and crossing bounds: (a) in 3 pages with crossing bound $b = 0$; (b) in 2 pages with crossing bound $b = 1$; (c) in 2 pages with crossing bound $b = 2$.

3. K_8 complete graph

K_8 complete graph can be embedded in 4 pages as shown in Figure 3.1. Book drawings of the complete graph K_8 in 3 pages for $b = 1$ and $b = 2$ are shown in Figure 5.3a and Figure 5.3b, respectively.

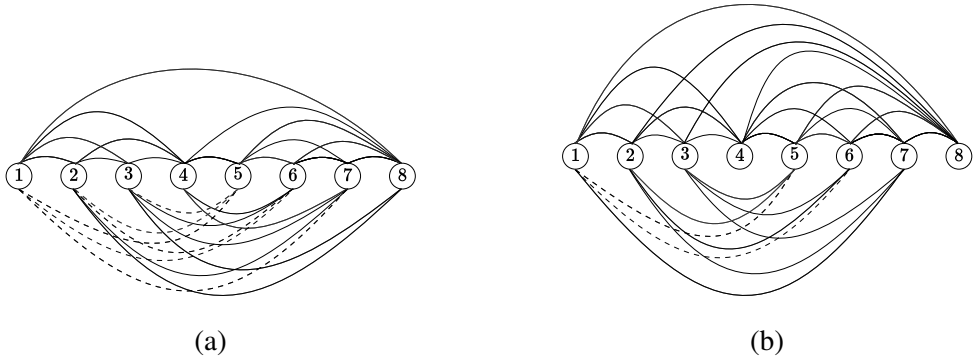


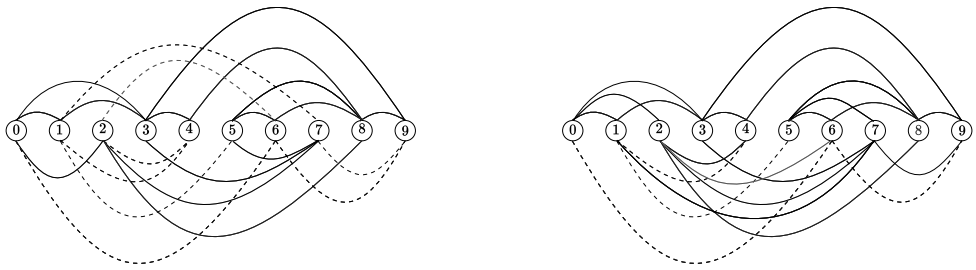
Figure 5.3: K_8 book drawing in 3 pages with different crossing bound: (a) $b = 1$; (b) $b = 2$

4. Generalized De Bruijn graph

An n-vertex k-regular generalized De Bruijn graph, denoted as $GDBG(n, k)$, consists of n vertices labelled from 0 to $n - 1$. Each vertex in the graph has

exactly k incoming edges and k outgoing edges. For any two vertices u and v, there is a directed edge from u to v if and only if:$v = (u \cdot k + \alpha) \mod n$where α is an integer such that $\alpha \in \{0, 1, \ldots, k-1\}$.

In this version, we replace directed edges with undirected edges and remove self-loops and parallel edges, transforming the graph into an undirected graph. Book drawings of Generalized De Bruijn graph $GDBG(10, 3)$ with $b = 0$ and $b = 1$ are shown in Figure 5.4a and Figure 5.4b.



(a) (b)

Figure 5.4: Generalized De Bruijn graph $GDBG(10, 3)$ book drawings with different page and crossing bounds: (a) in 4 pages with crossing bound $b = 0$; (b) in 3 pages with crossing bound $b = 1$.

5. Grid graph

A grid graph is a graph in which the vertices correspond to the points in a regular $m \times n$ rectangular grid, and edges connect vertices that are adjacent in the grid. Formally, the grid graph G has vertices (i, j), where $1 \leq i \leq m$ and $1 \leq j \leq n$, and an edge exists between two vertices (i, j) and (k, l) if and only if $|i - k| + |j - l| = 1$.

Any square grid can be embedded in two pages. Here, the book embedding of a square grid graph of size $4m \times n$ on two pages as illustrated in Figure 5.6. Vertices in order from 1 to 4mn place as shown in Figure 5.6. A row-wise ordering is assigned to the grid graph, created as $\{1, 2, \ldots, n\}, \{2n, 2n - 1, \ldots, n + 1\}, \ldots, \{4mn, 4mn - 1, \ldots, 4mn - n + 1\}$, as illustrated in Figure 5.5.

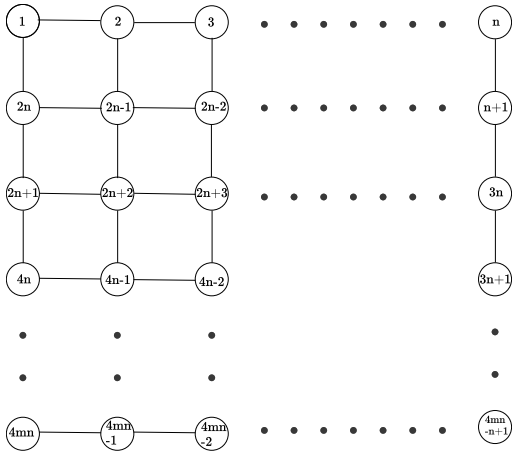


Figure 5.5: Square Grid graph

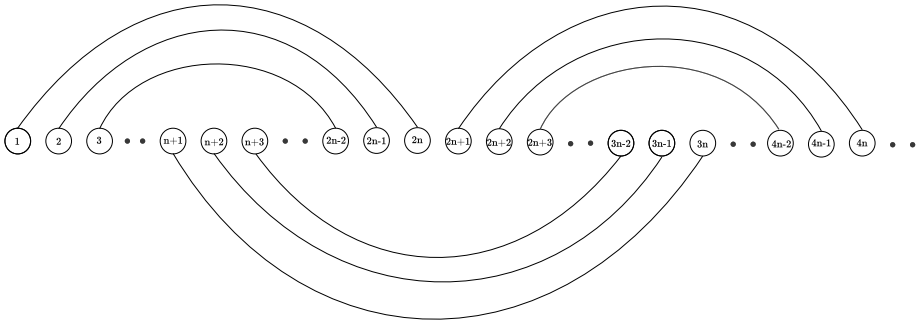


Figure 5.6: Grid graph of $4m \times n$ book embedding in 2 pages. First page contain all edges with $df(e) = 1$ and $((j-2)n+i+1, jn-i)$ where $j = \{2, 4, 6 \ldots, 4m\}$ and $i = \{0, 1, \ldots, n-2\}$. Second page contain edges $((j-2)n+i+1, jn-i)$ where $j = \{3, 5 \ldots, 4m-1\}$ and $i = \{0, 1, \ldots, n-2\}$.

6. Tutte graph

The Tutte graph is a 3-regular graph with 46 vertices and 69 edges, named after W. T. Tutte. Our algorithm embeds the Tutte graph in 3 pages according to the order given in Figure 5.7. If there is an edge between $(i, i+1)$ for $i = 1, 2, \ldots, 45$, it is placed on Page 1. The edges on each page are as follows:

- **Page 1**: (3, 45), (5, 39), (7, 10), (30, 34), (41, 45), (6, 25), (4, 40), (28, 35), (26, 36), (17, 20), (14, 20), (13, 21), (1, 46)
- **Page 2**: (1, 12), (22, 31), (23, 29), (32, 37), (38, 42), (37, 46), (24, 27), (33, 36), (19, 21), (43, 46)

- **Page 3**: (8, 18), (2, 44), (9, 16), (11, 15)

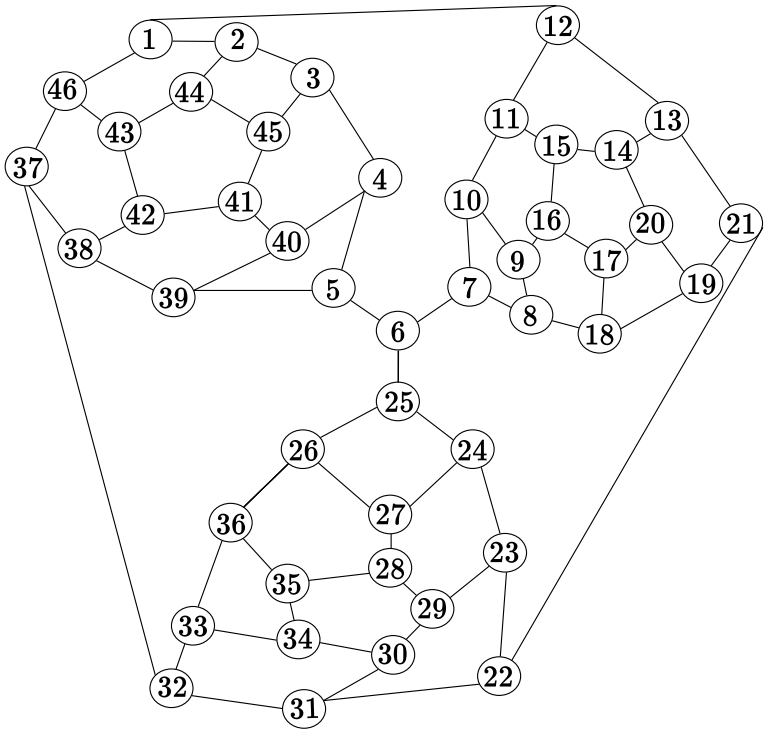


Figure 5.7: Tutte graph

Chapter 6

Experimental Result

In this section, we conduct experiments on random regular Hamiltonian graphs and random graphs generated using the Erdős–Rényi model. A random regular Hamiltonian graph is a graph that is regular and contains a Hamiltonian cycle. Random graphs are generated using the Erdős–Rényi model $G(n, p(n))$, where n is the number of vertices, and two vertices are connected by an edge with probability $p(n) = \frac{c\log(n)}{n}$. It has already been proven that if $c > 1$, then the graph is connected with high probability [12]. Therefore, in our experiment, we set $p(n) = \frac{2\log(n)}{n}$. In this model, the expected number of edges in the graph is given by: $\binom{n}{2} \cdot p = \frac{n(n-1)}{2} \cdot \frac{2\log(n)}{n} = (n-1) \cdot \log(n)$.

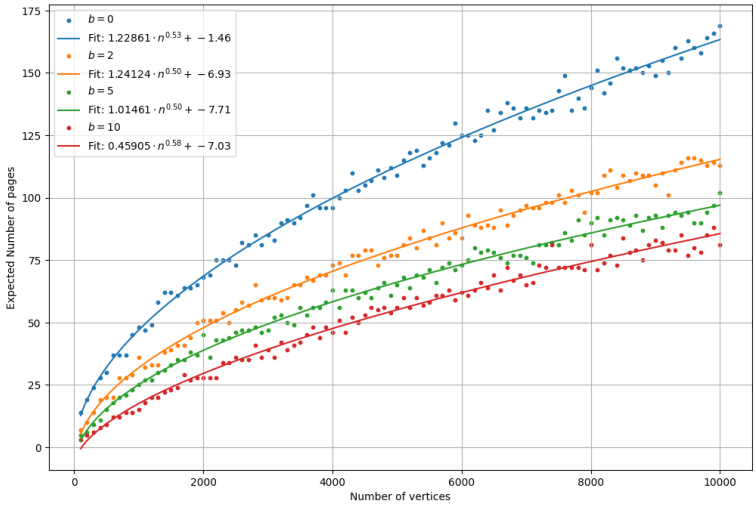
Using our algorithm, we calculate the expected number of pages required for the random graphs with different numbers of vertices. In this experiment, for random regular Hamiltonian graphs, the vertices are arranged along a spine according to the Hamiltonian cycle. For random graphs, the vertices are arranged along a spine randomly. Figure 6.1 shows the experimental results for random regular Hamiltonian graphs, while Figure 6.2 presents the results for random graphs generated using the Erdős–Rényi model.

In our study, we analysed the relationship between the number of vertices and the expected number of pages for random graphs with random vertex ordering. Using experimental data and curve fitting, we found that the expected number of pages without crossing can be approximated by the power-law model: $5.03974 \cdot n^{0.63} + -68.41$. This model provides a precise estimate of the page number trend as n increases. For a crossing bound of $b = 2$, the expected number of pages

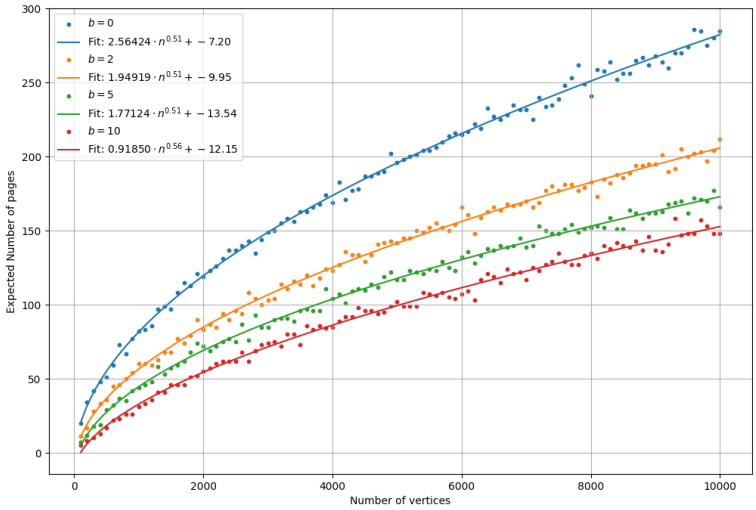
required, based on experiments, follows the model: $4.444146 \cdot n^{0.62} + -59.16$. Similarly, the expected number of pages for random regular Hamiltonian graphs was derived through experimental analysis and curve fitting, with the results summarised in Table 6.1.

Degree of vertices	**Crossing bound per edge b**	**Expected page number for n vertices**
3	0	$1.22861 \cdot n^{0.53} + -1.46$
3	2	$1.24124 \cdot n^{0.50} + -6.93$
3	5	$1.01461 \cdot n^{0.50} + -7.71$
3	10	$0.45905 \cdot n^{0.58} + -7.03$
4	0	$2.56424 \cdot n^{0.51} + -7.20$
4	2	$1.94919 \cdot n^{0.51} + -9.95$
4	5	$1.77124 \cdot n^{0.51} + -13.54$
4	10	$0.91850 \cdot n^{0.56} + -12.15$

Table 6.1: Expected page number for graphs with varying degrees of vertices and crossings per edge.



(a) Relationship between the number of vertices ($x-axis$) and the number of pages ($y-axis$) for 3-regular Hamiltonian graphs. Includes the expected number of pages derived from the experiment



(b) Relationship between the number of vertices ($x-axis$) and the number of pages ($y-axis$) for 4-regular Hamiltonian graphs. Includes the expected number of pages derived from the experiment

Figure 6.1: Experimental results for random regular Hamiltonian graphs: (a) Number of vertices vs. pages for 3-regular graphs, (b) Number of vertices vs. pages for 4-regular graphs

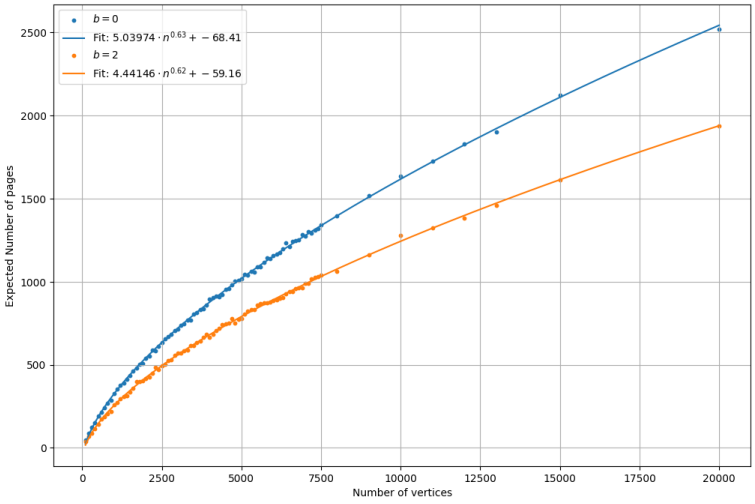


Figure 6.2: Relationship between the number of vertices ($x - axis$) and the number of pages ($y - axis$) for a random graph. Includes the expected number of pages with $b = 0$ and $b = 2$ derived from the experiment

Chapter 7

Conclusion and Future Scope

In this work, we address the problem of creating a book drawing for any graph G in $O(m^2)$ time, where m is the number of edges in the graph. Our algorithm produces a book drawing in which the number of crossings per edge is bounded by b, along with a ranking of the edges as parameter. The page assignment algorithm distributes the edges of the graph across the pages of the book such that the crossings per edge remain bounded by b. Finally, we present experimental results on random graphs and provide examples to demonstrate how our algorithm distributes edges across pages.

This work advances the study of book drawings by providing a scalable algorithm that operates within polynomial time and addresses practical constraints on crossings. The approach is particularly relevant for applications in areas such as graph visualization, network design, and VLSI layout optimization, where minimizing crossings and page usage is critical. Our contributions offer both theoretical insights and practical tools for constructing efficient book drawings for arbitrary graphs.

There are several directions in which this work can be extended. One important direction is to study how to draw graphs in a book when the order of the vertices is not already given. Choosing the best order can help reduce the number of pages or edge crossings, but finding such an order is a more difficult problem.

Another possible extension is to look at how we can remove a small number of edges from a graph to make it outer-b-planar (that is, drawable on k pages with

crossing bound b when all vertices are on the outer face). This can be studied using tools from parameterized complexity, by checking whether the problem becomes easier when certain graph properties (like treewidth) are small.

We can also try to design exact algorithms to count the number of edge crossings in graphs where the order of the vertices is already fixed. Even though this is a hard problem, it might be possible to solve it efficiently for special types of graphs.

Lastly, graph width measures like treewidth could be very useful in book drawing problems. For example, we can use dynamic programming on tree decompositions to place edges in pages more effectively, especially when the graph has a simple structure.

Bibliography

[1] Akanksha Agrawal, Sergio Cabello, Michael Kaufmann, Saket Saurabh, Roohani Sharma, Yushi Uno, and Alexander Wolff. Eliminating crossings in ordered graphs, 2024. URL: https://arxiv.org/abs/2404.09771, arXiv:2404.09771.

[2] A. Author. 1-planar graphs are not minor-closed. In *Graph Theory Notes*, 2025.

[3] B. Author. Book embeddings of 1-planar graphs. *Discrete Geometry Journal*, 2025.

[4] C. Author. On the book thickness of 1-planar graphs. In *Proceedings of the Graph Drawing Conference*, 2025.

[5] F. Author and G. Contributor. Book embeddings of 1-planar graphs. In *Proceedings of the International Symposium on Graph Drawing*, 2024.

[6] H. Author and I. Collaborator. Extending yannakakis' level technique to 1-planar graphs. In *Proceedings of the Graph Theory Workshop*, 2024.

[7] Frank Bernhart and Paul C Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. URL: https://doi.org/10.1016/0095-8956(79)90021-2.

[8] Sujoy Bhore, Robert Ganian, Fabrizio Montecchiani, and Martin Nöllenburg. Parameterized algorithms for book embedding problems. *Journal of Graph Algorithms and Applications*, 24, 12 2020. URL: https://doi.org/10.7155/jgaa.00526.

[9] Tomasz Bilski. Optimum embedding of complete graphs in books. *Discrete mathematics*, 182(1-3):21–28, 1998. URL: https://doi.org/10.1016/S0012-365X(97)00131-3.

[10] Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graphs in books: A layout problem with applications to vlsi design. *SIAM J. Algebraic Discrete Methods*, 8(1):33–58, jan 1987. URL: https://doi.org/10.1137/0608002.

[11] Vida Dujmović and David R. Wood. Treewidth and book thickness. *Journal of Algorithms*, 51(2):168–182, 2004.

[12] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[13] Joseph L. Ganley and Lenwood S. Heath. The relationship between treewidth and book thickness. *SIAM Journal on Discrete Mathematics*, 7(4):601–613, 1994.

[14] Paul C Kainen. Some recent results in topological graph theory. In *Graphs and Combinatorics: Proceedings of the Capital Conference on Graph Theory and Combinatorics at the George Washington University June 18–22, 1973*, pages 76–108. Springer, 2006. URL: https://doi.org/10.1007/BFb0066436.

[15] Yunlong Liu, Jie Chen, and Jingui Huang. Parameterized algorithms for fixed-order book drawing with bounded number of crossings per edge. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12577 LNCS, 2020. URL: https://doi.org/10.1007/978-3-030-64843-5_38.

[16] Seth Malitz. A bound on the book thickness of graphs with a given number of edges. *Journal of Graph Theory*, 17(4):435–441, 1994.

[17] Seth Malitz. Graphs of bounded genus have bounded book thickness. *Discrete Mathematics*, 146(1–3):255–262, 1995.

[18] L. Martinez and M. Wang. Structure and drawings of optimal 2-planar graphs. *Discrete Mathematics and Theoretical Computer Science*, 29(4):215–230, 2021.

[19] N. Patel and O. Lee. Book thickness of *h*-planar graphs: Bounds and algorithms. *Algorithmica*, 82(1):90–115, 2020.

[20] D. Researcher. A survey on *h*-planar graphs. *Journal of Graph Embeddings*, 2024.

[21] E. Researcher. Planarity with edge crossings: An updated overview. *Algorithmic Graph Theory*, 2024.

[22] Neil Robertson and Paul D. Seymour. Graph minors. xvi. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003.

[23] J. Smith and K. Johnson. Caging edge crossings in 1-planar graphs. *Journal of Graph Drawing and Visualization*, 25(2):123–138, 2020.

[24] Mihalis Yannakakis. Four pages are necessary and sufficient for planar graphs. *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 104–108, 1986.

[25] Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1):36–67, 1989.