

To the Graduate Council:

I am submitting herewith a dissertation written by Cheng Chen entitled "Novel Graph Algorithms for Life Science Applications." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Michael A. Langston, Major Professor

We have read this dissertation
and recommend its acceptance:

Dr. Michael A. Langston

Dr. Margaret Staton

Dr. Michael W. Berry

Dr. David Icové

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

To the Graduate Council:

I am submitting herewith a dissertation written by Cheng Chen entitled “Novel Graph Algorithms for Life Science Applications.” I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Michael A. Langston, Major Professor

We have read this dissertation
and recommend its acceptance:

Dr. Michael A. Langston

Dr. Margaret Staton

Dr. Michael W. Berry

Dr. David Icové

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Novel Graph Algorithms for Life Science Applications

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Cheng Chen
December 2025

© by Cheng Chen, 2025
All Rights Reserved.

I dedicate this dissertation to my parents, who carried me through every challenge and triumph of this journey. I could not have completed this without your unwavering love, support, and encouragement.

Acknowledgements

I am deeply grateful to my advisor, Dr. Michael A. Langston, for his constant guidance, support, and encouragement during my PhD. His deep knowledge, clear thinking, and patient teaching have greatly influenced the direction and quality of my dissertation. Dr. Langston's sharp insight, helpful feedback, and strong belief in my abilities helped me grow as a researcher. He always balanced high expectations with kindness and humor, which made working with him both rewarding and enjoyable.

I would also like to extend my sincere appreciation to my committee members—Dr. Margaret Staton, Dr. Michael W. Berry, and Dr. David Icové—for their valuable input, constructive suggestions, and ongoing support. Each of them brought unique perspectives and expertise that enriched the scope of my research. Dr. Staton's knowledge of plant biology and genomics helped bridge the gap between computational and biological sciences; Dr. Berry's foundational contributions to data analysis and mathematical modeling methods deepened my understanding of algorithmic techniques; and Dr. Icové's experience in applied computing and systems thinking challenged me to consider practical implications beyond theory. Their thoughtful feedback consistently pushed me to refine my ideas and elevate the quality of my work.

I sincerely thank my amazing team members, Haoran Niu, Yuping Lu, Stephen Grady, Levente Dojcsak, Daniel Mishler, Everett Rush, Viktor Zenkov, Tyson Beach, Kapidev Neupane, Aaron Bailey, and Levi Hochstetler, for their encouragement and teamwork. Special thanks go to my friends, Jingzhi Pang and Yuxin Fan, who have

shared this journey with me. Their camaraderie and words of encouragement have made this experience both fulfilling and enjoyable.

Finally, I am profoundly grateful to my parents for their love, patience, and unwavering belief in me. In my most difficult times, they were my solid support. Their constant support has been my greatest source of strength and inspiration throughout this challenging process.

Important Notice. The grammar, diction, and syntax of this dissertation have been substantially revised and improved with the aid of the AI Assist add-on to the Overleaf LaTeX editor. The final content was reviewed and approved by the author.

Abstract

Modern life sciences increasingly leverage graph-theoretic models to untangle complex biological relationships; however, existing algorithms often struggle with the scale and heterogeneity of life science data. This dissertation addresses these challenges by developing and applying advanced graph theory algorithms in key domains of the life sciences.

First, the dissertation introduces a novel link prediction algorithm, Triclique Augmentation, for multipartite biological networks, with a focus on disease-drug-protein interaction graphs. This algorithm integrates topological features with domain-specific information to infer new links in a tripartite network, facilitating drug repositioning discoveries. Its performance was evaluated using several widely adopted bioinformatics tools, all of which confirmed that the algorithm identifies biologically plausible associations. This method not only reduces the economic and temporal costs of pharmaceutical development, but also pioneers the use of multipartite graphs in bioinformatics—moving beyond traditional bipartite models to enhance prediction accuracy.

Second, the dissertation explores the structural properties of k -partite graphs, focusing on the enumeration of maximal k -partite cliques. To overcome limitations in existing algorithms, an improved method called MPCE is introduced, which enhances efficiency by eliminating non-viable candidates early in the search. Building on the Bron-Kerbosch algorithm, MPCE is designed for scalability and is validated on both synthetic and biological datasets.

Third, the dissertation presents the Power Gabriel Graph (PGG), a graph construction algorithm that extends the classical Gabriel Graph by incorporating a tunable power parameter. This enables flexible control over edge filtering and graph density. Unlike traditional global thresholding methods used in gene network construction, PGG uses local topological structure to make adaptive edge retention decisions—a rarely employed approach in this domain. Applied to gene expression data, PGG successfully constructs biologically meaningful co-expression networks. Theoretical properties of the resulting graphs are also formally established.

These contributions create a framework where advanced graph algorithms help make new discoveries in life sciences, showing how computer science and biology can benefit each other.

Table of Contents

1	Introduction	1
1.1	Graph Theory Background	1
1.2	Link Prediction in Graphs	2
1.3	K -partite Graph Background	4
1.4	Maximal Clique Enumeration Background	6
1.5	Graph Construction Background	7
1.6	Outline	8
2	Link Prediction in Multipartite Graphs with Application to Drug Repositioning Studies	10
2.1	Introduction	11
2.2	Methodology	13
2.2.1	Tripartite Graph Construction	13
2.2.2	Link Prediction	14
2.2.3	Nonadjacency Bound	15
2.3	Evaluating Disease-Drug Link Predictions	18
2.4	Evaluating Disease-Protein Link Predictions	19
2.5	Evaluating Drug-Protein Link Predictions	26
2.6	Discussion	29
3	Faster Maximal K-partite Clique Enumeration (MPCE)	31
3.1	Introduction	32

3.2	Algorithm	35
3.2.1	Pruning the Search Tree and Pseudo Code	37
3.2.2	Time and Space Complexity	39
3.3	Algorithm Performance Assessment	40
3.3.1	Algorithm Evaluation on Pseudorandomly Generated Graphs	41
3.3.2	Algorithm Evaluation on Real-World Data	52
3.4	Discussion	55
4	The Power Gabriel Graph	57
4.1	Introduction	58
4.2	Power Gabriel Graph	59
4.2.1	Key Properties of the Power Gabriel Graph	61
4.2.2	Time Complexity of Constructing a Power Gabriel Graph	63
4.3	Implementation and Testing	64
4.3.1	Datasets and Graph Construction	64
4.3.2	Graph Vertex Clustering Analysis	65
4.4	Discussion	70
5	Conclusions and New Research Directions	74
	Bibliography	78
A	Supplementary Analysis of Chapter 2 Experiments	95
A.1	Extended Analysis for Section 2.3	95
A.2	Extended Analysis for Section 2.4	96
A.3	Molecular Docking Protocol Validation	97
B	Datasets for Chapter 4 Experiments	101
	Vita	104

List of Tables

1.1	Vertex dissimilar score methods	8
2.1	Vertex and edge counts for the five tripartite subgraphs under study.	14
2.2	Link predictions (edge imputations) by link type and disease classification.	16
2.3	Molecular function enrichment by GO category and disease classification.	22
2.4	Binding energy metrics and Mann-Whitney test results.	29
3.1	Metrics of imbalanced test graphs	48
3.2	Running time and data metrics on BioSNAP tripartite graphs	55
4.1	Graph counts where one method outperforms another in SCR across power settings for <i>Homo sapiens</i> data	69
4.2	Graph counts where one method outperforms another in SCR across power settings for <i>Mus musculus</i> data	72
A.1	Best model RMSD between the docked and co-crystal ligands.	99
B.1	Summary of <i>homo sapiens</i> GEO data sets used in graph construction experiments	102
B.2	Summary of <i>Mus musculus</i> GEO datasets used in graph construction experiments	103

List of Figures

2.1	Typical edge recovery experience.	17
2.2	Mean number of shared adjacent proteins for diseases and drugs in highest ranked edges.	20
2.3	Mean number of shared adjacent drugs for diseases and proteins in highest ranked edges	27
3.1	Example of a tripartite graph and its tripartite cliques.	33
3.2	Example of two maximal bipartite cliques (green) in tripartite graphs	35
3.3	How the running time, the number of tripartite cliques, and the memory usage changed as the number of vertices in each partite set increases.	43
3.4	How the running time, the number of k -partite cliques, and the memory usage changed as the number of partite sets increases.	44
3.5	How the running time, the number of tripartite cliques, and the memory usage changed with the density.	47
3.6	Performance analysis on imbalanced k -partite graphs with uneven partite set sizes.	49
3.7	Performance analysis on imbalanced k -partite graphs with uneven inter-partite edge densities.	53
4.1	Histogram of spectral graph theory thresholds values of test graphs .	66

4.2	Mean significant clusters ratio (SCR) and number of clusters across different power values in <i>Homo sapiens</i> test data.	68
4.3	Mean significant clusters ratio (SCR) and number of clusters across different power values in <i>Mus musculus</i> test data.	71
A.1	Histograms of the BLAST bit score distributions.	98
A.2	Comparison of docked (blue) and co-crystalized (magenta) DB00679 from PDB ID 3TBG.	100

Nomenclature

E	Edges set
G	Graph
$N(v)$	Neighbourhood of a vertex v
$P(v)$	The partition set where vertex v is located
u, v, w	Single vertex examples
V	Vertices set

Chapter 1

Introduction

This chapter provides the background for my Ph.D. research, beginning with a brief history of graph theory and relevant theoretical foundations. It then presents the motivation for the research, outlines key challenges, summarizes the main contributions of the research conducted during my Ph.D. studies, and offers an overview of the structure of the dissertation.

1.1 Graph Theory Background

A graph $G(V, E)$ is a structure consisting of a set of vertices V and a set of edges E , where each edge connects a pair of vertices. Graphs are widely used in various fields, as vertices can represent instances and edges can represent the relationships between them. Graph theory, the mathematical study of graphs, holds both theoretical and practical significance, contributing to advancements across numerous disciplines.

Graph theory originated with the work of Leonhard Euler in the eighteenth century, who formulated the famous Königsberg bridge problem (1). Euler's solution is considered the first theorem in graph theory and laid the groundwork for the field. Since then, graph theory has evolved significantly. Over the years, graph theory has become foundational in numerous areas of mathematics and computer

science, particularly in combinatorics (2), algorithms (3), and network theory (4). Its applications have expanded into an even broader range of fields.

Graphs can adopt various structures and configurations depending on the application’s requirements. Directed graphs, where each edge has a specific direction, are useful for modeling asymmetric relationships, such as signal processing in a network (5), asymmetric drug-drug interactions(6), and task dependencies in a scheduling system (7). Weighted graphs, where each edge is assigned a numerical value or weight, are commonly used in scenarios like road networks where the weight might represent travel time between locations (8), brain connectivity analysis (9), and keyword search in query processing (10).

1.2 Link Prediction in Graphs

The current landscape of link prediction algorithms for graphs is diverse, spanning from classical similarity-based methods to sophisticated machine learning models. The similarity-based methods including:

- Jaccard Coefficient (11): Measures the similarity between the sets of neighbors of two nodes, which is particularly useful in sparse k -partite graphs.
- Adamic/Adar Index (12): Weighs the shared connections between two nodes, giving more weight to rare connections, which is effective in highlighting significant but less obvious links.
- Preferential Attachment (13): Assumes that nodes with higher degrees are more likely to form links, a principle that applies broadly across different types of networks.

The probabilistic models based link prediction including:

- Probabilistic Relational Models (PRMs) (14): Use the probabilistic dependencies between attributes of node pairs to predict links.

- Latent Space Mapping(15): Position nodes in a latent space where distances in this space reflect the probability of link formation. These models are useful for capturing unobserved heterogeneity among nodes.

Matrix factorization methods decompose the adjacency matrix of the graph into lower-dimensional latent factors for nodes, which can then be used to predict missing links (16):

- Singular Value Decomposition (SVD) (17): Identifies latent factors in the adjacency matrix that explain observed links and predict new ones, effectively handling the sparsity and high dimensionality of k -partite graphs.
- Non-negative Matrix Factorization (NMF) (18): Decomposes the adjacency matrix into non-negative factors, making this method suitable for graphs where the relationships have inherently non-negative meanings (e.g., ratings, frequencies).

Advanced machine learning techniques, such as random forests (19) and gradient boosting machines (20), have been successfully adapted to enhance the accuracy and scalability of link prediction in graphs. These methods, known for their robustness in handling complex, noisy data, leverage an ensemble of decision trees to model non-linear relationships across diverse datasets, making them particularly suited for the structured yet heterogeneous nature of k -partite graphs. Concurrently, graph neural networks (GNNs) (21; 22) have increasingly become central to advancements in link prediction. GNNs, by design, exploit the rich relational information embedded within the graph structure, enabling them to outperform traditional machine learning models on tasks where topology plays a crucial role. Their ability to learn node representations that effectively capture both local graph structures and node features has proven essential in predicting potential links with high precision. As these technologies evolve, they collectively push the boundaries of what is possible in the domain of network analysis, offering more nuanced and scalable solutions for predicting links in k -partite and other complex network structures.

In Chapter 2, I present a novel link prediction algorithm that leverages the structural properties of the network, rather than relying on detailed vertex or edge attributes. This makes the method particularly robust and efficient in scenarios involving heterogeneous data sources or sparse, inconsistent attribute information. By focusing on the underlying graph topology—especially connectivity patterns in subgraphs such as k -partite cliques—the algorithm effectively predicts potential links, demonstrating strong performance in complex network environments where traditional feature-based approaches may struggle.

1.3 K -partite Graph Background

Chapters 2 and 3 of this dissertation focus on a specific graph structure— k -partite graphs—which are particularly well-suited for analyzing heterogeneous data. A graph $G = (V, E)$ is said to be k -partite if there exists a partition of V into k distinct sets, and for every edge $(u, v) \in E$, vertices u and v belong to different sets. If $k = 2$, the graph is referred to as a bipartite graph; if $k = 3$, it is a tripartite graph, and so on. A k -partite clique is a subset of a k -partite graph which has vertices from every partition set and where every vertex in one partition is connected to every vertex in the other partitions. If a k -partite clique cannot be extended by adding an adjacent vertex, it is called maximal k -partite clique. The one with largest size among all the maximal k -partite clique is called maximum k -partite clique.

The structural properties of k -partite graphs emphasize intergroup relationships, making them applicable across a broad range of domains, including:

- **Biological Networks:** k -partite graphs are used to model complex biological systems, such as protein-protein interactions (23), gene-disease associations (24), altered brain connectivity networks (25), and metabolic networks (26).
- **Social Networks:** In social network analysis, k -partite graphs are used to represent relationships among multiple types of entities such as users, content,

and topics (27) or entities with different characteristics such as users with different background (28). This is particularly relevant for collaborative filtering (27) and recommendation systems (29).

- Database and Information Retrieval: In database theory, k -partite graphs can represent relations between different types of entities, facilitating efficient querying and retrieval processes (30). Similarly, in information retrieval, k -partite graphs help in document clustering (31).
- Network Efficiency: The k -partite could be used to optimize the placement of Access Control List (ACL) policies within hybrid SDN systems, reducing unnecessary traffic and minimizing the number of ACL rules (32). This enhances network performance and efficiency by improving traffic management and ensuring policies are implemented in optimal locations, thus directly contributing to overall network efficiency.
- Textile engineering: K -partite graphs are applied in textile engineering to optimize the routing of thread spools on a braiding machine to prevent collisions (33). This is achieved through modeling the problem as a maximal clique issue, where each spool represents a vertex, and compatible routing paths are edges.
- Collaborative Filtering and Recommendation Systems: k -partite cliques can be used to find groups of items that a set of users commonly likes or interacts with, enhancing recommendation algorithms (34). For example, in a user-item-tag k -partite graph, a k -partite clique would represent a perfect subset where users, items, and tags are all interconnected, suggesting high relevance and potential for recommendation.
- Data Integration: In scenarios where different datasets need to be integrated, k -partite cliques provide a method to identify strong linkages across diverse datasets.

1.4 Maximal Clique Enumeration Background

The problem of maximal clique enumeration is a foundational topic in the field of graph theory and computational mathematics, drawing significant attention due to its broad range of applications across various scientific and industrial domains (28; 33; 35; 36). Maximal cliques are crucial for understanding the dense components of graphs, which can represent tightly-knit groups within networks. These dense components are essential for network analysis in different areas such as social sciences (37) and biology (25; 38; 39). For instance, in social network analysis, maximal cliques can represent groups of people who all know each other; in biological networks, they can indicate functional modules or protein complexes.

The computational challenge of enumerating all maximal cliques in a general graph is considerable, as the problem is $\mathcal{NP}$ -hard (40). The number of maximal cliques in a graph can grow exponentially with the number of vertices (41), and therefore, efficient algorithms are crucial for practical applications. The classic algorithm by Bron and Kerbosch (BK) (42) is a well-known recursive method for finding all maximal cliques using a backtracking approach, which is often preferred for its simplicity and performance on many types of graphs. Due to its simplicity and efficiency, the BK algorithm has become the basis for numerous subsequent studies and algorithmic improvements in maximal clique detection.

Over the years, various enhancements and optimizations have been proposed to improve the practical performance of maximum clique enumeration algorithms such as parallel processing (40; 43; 44; 45; 46), and heuristic algorithms (47). Each approach offers a different balance of efficiency and computational complexity, tailored to specific types of graphs and application needs. In contrast, the problem of maximal clique enumeration—especially maximal k -partite clique enumeration—has received comparatively less attention in the literature. In Chapter 3, I present an enhanced maximal k -partite clique enumeration algorithm and demonstrate its performance on both synthetic and real-world graphs.

1.5 Graph Construction Background

Graphs are fundamental data structures widely employed across diverse research domains. However, the required graphs are not always readily available, and researchers often need to construct them from raw data. The quality of the constructed graph representations has a direct impact on the performance of graph-based algorithms, making graph construction a critical pre-processing step that can significantly influence the success of subsequent analyses. Most graph construction algorithms rely on proximity-based methods, which establish edges between vertices using either similarity (or weights) or dissimilarity (or distances) measures. Correlation coefficients, such as Pearson and Spearman, are commonly employed to compute edge weights, while their complementary values (e.g., $1 - |\text{correlation}|$) can be utilized to represent vertex dissimilarity relationships.

Table 1.1 presents commonly used methods for computing dissimilarity between vertices. Let A and B be vectors corresponding to two vertices, where a_i and b_i denote the i -th components of A and B , respectively—representing, for example, the values of the i -th feature. The term $\text{cov}(A, B)$ denotes the covariance between vectors A and B , while σ_A and σ_B represent the standard deviations of vectors A and B , respectively. From the equations in the Table 1.1, we can see that dissimilarity scores based on Euclidean and Manhattan distances can be heavily influenced by very large values in the vector. Therefore, when using Euclidean or Manhattan methods to measure dissimilarity, the data need to be standardized by removing the mean and scaling to unit variance in some studies.

After obtaining edge similarity or dissimilarity scores, a straightforward approach to graph construction is to apply a threshold—retaining only those edges with similarity scores above the threshold or dissimilarity scores below it. The study by (48) provides a detailed discussion of various thresholding algorithms used in gene data analysis and compares their performance. In Chapter 4, I propose a novel graph construction algorithm that determines edge retention by comparing each

Table 1.1: Vertex dissimilar score methods

Dissimilar Method	Dissimilar Equation
Euclidean Distance Score	$\sqrt{\sum_{i=1}^m (a_i - b_i)^2}$
Manhattan Distance Score	$\sum_{i=1}^m a_i - b_i $
Complementary of the Pearson Correlation Score	$1 - \left \frac{\text{cov}(A, B)}{\sigma_A \sigma_B} \right $
Complementary of the Spearman Correlation Score	$1 - \left \frac{\text{cov}(\text{rank}(A), \text{rank}(B))}{\sigma_{\text{rank}(A)} \sigma_{\text{rank}(B)}} \right $

edge’s weight with those of its adjacent edges, rather than relying on a single global threshold. This local, context-aware criterion enables adaptive edge selection that reflects the structural characteristics of each node’s neighborhood. I integrate this algorithm with the global thresholding approach in the context of gene data analysis, aiming to construct more reliable graphs by leveraging the strengths of both methods.

1.6 Outline

This dissertation is structured into five main chapters, with Chapters 2 to 4 each exploring a distinct facet of graph algorithms and their applications in life science research. In Chapter 2, I introduce a new link prediction algorithm and its application in disease-drug-protein tripartite graphs. I also present the validation results of the predicted links using various bioinformatics analyses. In Chapter 3, I detail my enhanced k -partite clique enumeration algorithm, which was evaluated on both artificially generated pseudorandom graphs and real-world data. Chapter 4 discusses a novel graph construction algorithm that relies on the relative weights of edges compared to adjacent edges, rather than on their absolute values. Finally, in Chapter 5, I summarize my work, highlight my contributions, and discuss potential directions for future research. All graphs used in this dissertation are simple (no multiple edges between any two vertices), unweighted, undirected, finite (with a limited number of

vertices and edges), and static (the set of vertices and edges remains unchanged over time) graphs, which are commonly encountered in many applications.

Chapter 2

Link Prediction in Multipartite Graphs with Application to Drug Repositioning Studies

*†

Developing new ethical drugs is exceedingly expensive in terms of both time and resources. A single drug can take up to a decade to bring to market, with costs soaring to over a billion dollars. Drug repositioning has thus become an attractive alternative to the development of new compounds, with growing interest in the use of in silico repositioning predictions. Bipartite graphs and efficient biclique enumeration algorithms can be used to study drug-protein or other pairwise crucial interactions. Extensions of this approach to datasets with three or more divergent data types have been hobbled, however, by a lack of effective analytics. To address this shortcoming, a highly innovative and efficient graph theoretical technique is introduced to impute

*This chapter is based on the manuscript “*Link Prediction in Multipartite Graphs with Application to Drug Repositioning Studies*,” by Cheng Chen, Stephen K. Grady, Levente Dojcsak, Sally R. Ellingson, and Michael A. Langston, currently under revision at *IEEE Transactions on Computational Biology and Bioinformatics*.

†A preliminary version of a portion of this paper was presented at the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, which was held online in August 2021

potential edges (links) in an arbitrary multipartite graph. The utility of this method is demonstrated on five tripartite graphs, each comprised of three partite sets, one each for drugs, diseases, and gene products of interest, and with interpartite edges denoting known interactions or associations. Evidence for the reliability of imputed edges is also reported.

2.1 Introduction

Traditional drug discovery and development entail substantial costs, routinely requiring funding of over a billion USD and development times of over a decade(49; 50). Moreover, the proportion of compounds that make it to market is low, further diminishing return on investments. In an effort to mitigate these costs, computer-aided drug design(51) has become an attractive alternative. A particularly promising framework for this is drug repositioning, which encompasses methods for determining new medical indications for existing FDA approved pharmaceuticals, thereby circumventing the time and finances needed to develop novel compounds. Compared to the billions typically required to develop a new drug, bringing a repurposed drug to market is estimated to cost an average of \$300 million (52). One of the oldest and best-known examples of drug repositioning is Aspirin. It was first named by the German pharmaceutical giant Bayer near the end of the 19th century and prescribed to treat fever and inflammation (53). It was repositioned in the 1980s for antiplatelet aggregation (54), and suggested only recently as a colorectal cancer preventative (55). Unfortunately, for afflictions such as Alzheimer’s disease, entire decades may pass with no new drug candidate approvals (54). This makes repurposing an especially appealing strategy for timely and efficacious treatment of disease.

A common approach to drug repositioning is based on leveraging publicly available databases containing known chemical structures to predict possible drug-target interactions. Computer-assisted protein-ligand binding analysis is widely included

in both disease-protein relationship studies and drug-protein studies(50). This technique was used, for instance, to analyze possible binding sites between 200 antiviral phytocompounds and the SARS-CoV-2 main protease to determine potential inhibitors (56).

Graph theory too has found a natural role in drug repositioning. In this setting, biological entities are represented by vertices, and their relationships are denoted by edges (aka links). Thus, the process of determining these relationships is often framed as a link prediction task. Examples of this approach include the use of neighborhood similarity(57; 58), network diffusion(59), and matrix projection to generate drug-target interaction predictions(60). Melding highly heterogenous data can sometimes result in more accurate predictions, although this poses its own set of challenges(61) and prompts the use of multipartite graphs. Specifically, a k -partite graph is one whose vertices can be partitioned into k nonempty partite sets so that every edge has its endpoints in different sets(41). Bipartite graphs, those for which $k = 2$, have previously been used in drug repositioning studies(58; 59; 62; 63), but little consideration has thus far been seen for scaling to larger values of k .

In this paper, we address this issue with the presentation of a scalable graph theoretic algorithm that can be used to predict edges between an arbitrary number of partite sets. We test the utility of this method on a tripartite graph, whose partite sets respectively denote diseases, drugs, and gene products, and whose interpartite edges represent known interactions and associations. Although the primary focus of this paper is on emergent combinatorial tools for link predictions, we also apply well established bioinformatics methodology to estimate the potential validity of our results.

The remainder of this paper is organized as follows. In the next section, we explain how to construct tripartite graphs from heterogenous data, work through a handful of well-chosen examples, and describe the algorithm we use for link prediction. In Sections 2.3, 2.4, and 2.5, respectively, we provide supplemental evidence to support the pertinence of the disease-drug, disease-protein, and drug-protein links that our

method predicts. In the final section, we summarize our work, review its limitations, and suggest possible avenues for future study.

2.2 Methodology

2.2.1 Tripartite Graph Construction

All graphs used in this paper are finite, simple, and undirected. A k -partite clique in a k -partite graph is a set of vertices that induces a complete k -partite subgraph.

We began with three bipartite networks (undirected graphs) maintained by the Stanford Biomedical Network Dataset Collection (BioSNAP) (64). These were ChG-Miner (drugs and gene products), DCh-Miner (drugs and diseases), and DG-Miner (diseases and gene products). When there is no confusion, we will merely refer to gene products from these sources as proteins.

Data in ChG-Miner contains known interactions between proteins and drugs or nutraceuticals on the US market. Nutraceuticals are distinct from ethical, also known as prescription, drugs in that they are usually taken to supplement health rather than to treat disease. BioSNAP combines the two, however, into a single category that we will simply term drugs. Associations in this dataset were derived from both DrugBank (65) and the Gigascale multimodal biological network (MINER) (66). DCh-Miner contains known associations between diseases and drugs. Prescriptions were extracted from the Comparative Toxicogenomics Database (CTD) (67) and MINER. Finally, DG-Miner contains known associations between diseases and proteins ascertained from CTD, MINER, and the Online Mendelian Inheritance in Man (OMIM) (68) databases.

We constructed a tripartite graph by first taking the union (69) of the three aforementioned bipartite graphs. We then removed any vertex that was not adjacent to at least one vertex from each of the two partite sets in which it did not reside. From this, we extracted five tripartite subgraphs, each anchored by a family of

Table 2.1: Vertex and edge counts for the five tripartite subgraphs under study.

Graph Metrics	Graph MeSH Categories				
	Eye Disease	Heart Disease	Hemic & Lymphatic	Infections	Wounds & Injuries
Disease Vertices	88	40	79	150	71
Drug Vertices	844	963	925	814	885
Protein Vertices	1987	1992	1991	1977	1986
Disease-Drug Edges	5539	6630	9272	23068	6090
Disease-Protein Edges	59821	33524	74530	115283	31225
Drug-Protein Edges	5402	5826	5673	6232	5656

disease vertices categorized by their Medical Subject Headings (MeSH) codes (70) along with their drug and protein neighbors. The MeSH categories we chose were eye diseases (C11), heart diseases (C14.280), hemic and lymphatic diseases (C15), infections (C01), and wound injuries (C26). Although no vertex chosen for its disease classification was contained in two or more of these five subgraphs, their neighbors sometimes were, which was both expected and acceptable because this simple form of graph decomposition does not require disjointness. This list of disease categories is of course arbitrary. It was chosen merely as a proof of concept and to represent a highly diverse set of tissues and afflictions. Vertex and edge counts for these five subgraphs are listed in Table 2.1. Densities ranged between two and seven percent.

2.2.2 Link Prediction

We generated all maximal tricliques using the k -partite clique enumeration algorithm described in (41). To ensure that disease, drug, and protein data were all at least marginally represented, we retained for analysis only those tricliques with at least three vertices in each partite set. Otherwise, we note as examples that only one or two diseases restricts the opportunity to study pathological similarity, only one or two drugs limits the potential for pharmaceutical retargeting, and only one or two

proteins hampers our ability to pinpoint molecular pathways. Due to imbalance, no maximum triclique withstood this test.

Let C denote a triclique satisfying this criterion. We next identified any non-triclique vertex, v , that was adjacent to all but a handful of C ’s vertices not in v ’s partite set. Note that v must be nonadjacent to at least one such vertex, since otherwise v would be an element of C . On the other hand, too many nonadjacencies weaken v ’s potential utility in the context of link prediction. For this study, we allowed no more than two nonadjacencies based on extensive bootstrapping analysis, which we will detail in the next subsection. Edges (aka links) were then imputed between v and these formerly nonadjacent vertices of C . We shall henceforth term this procedure “triclique augmentation” and hasten to note that it does not consider non-clique adjacencies, nor is it designed to preserve density. Thus, triclique augmentation should not be confused with the superficially similar paraclique algorithm (71), which is intended mainly for clustering applications in which there is a need to ameliorate the effects of noise. Link prediction counts are listed in Table 2.2.

2.2.3 Nonadjacency Bound

We use the term “nonadjacency bound” to denote an upper limit on the number of triclique nonadjacencies permitted by the link prediction algorithm just described. To determine a best limit over these data, we first created a suite of test graphs by pseudorandomly deleting 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of the edges from each of the five tripartite disease-specific graphs we had previously created. We then defined the “recovery rate” as the number of recovered edges divided by the number of removed edges. While this ensures a direct measure of link prediction’s success, it can only increase with the nonadjacency bound. Therefore, to accompany the recovery rate, we defined the “recovery percentage” as the number of recovered edges divided by the number of predicted edges. This ratio is of course a

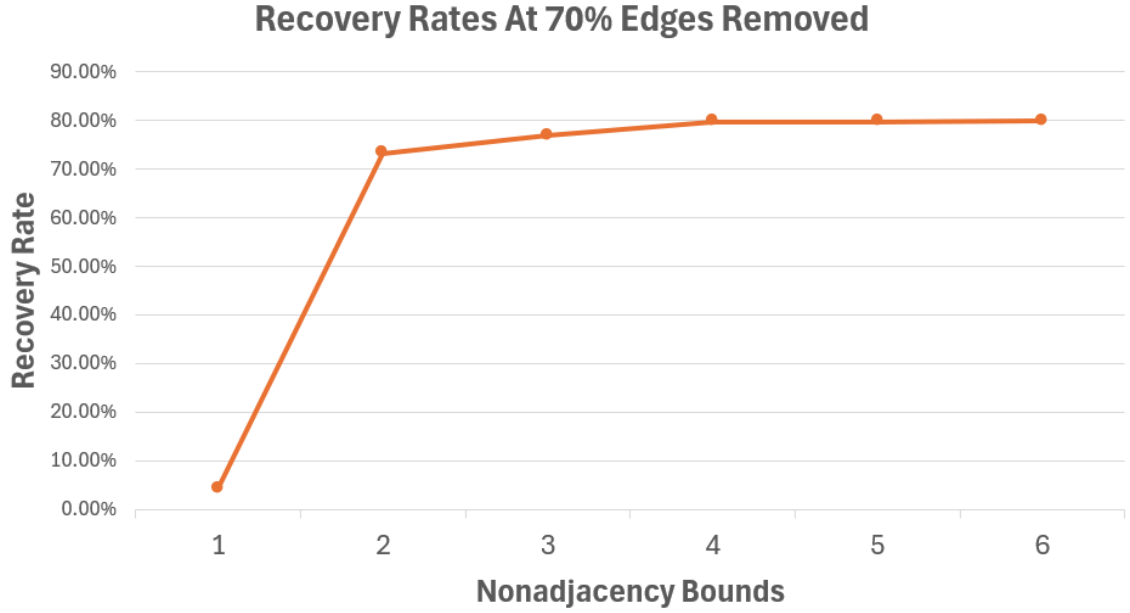
Table 2.2: Link predictions (edge imputations) by link type and disease classification.

Graph Computations	Graph MeSH Categories				
	Eye Disease	Heart Disease	Hemic & Lymphatic	Infections	Wounds & Injuries
Tricliques Meeting Inclusion Criterion	16409	18071	108249	20750	10239
Disease-Drug Link Predictions	5958	3722	9218	4998	3404
Disease-Protein Link Predictions	418	242	812	566	540
Drug-Protein Link Predictions	11207	31886	24457	11195	26372

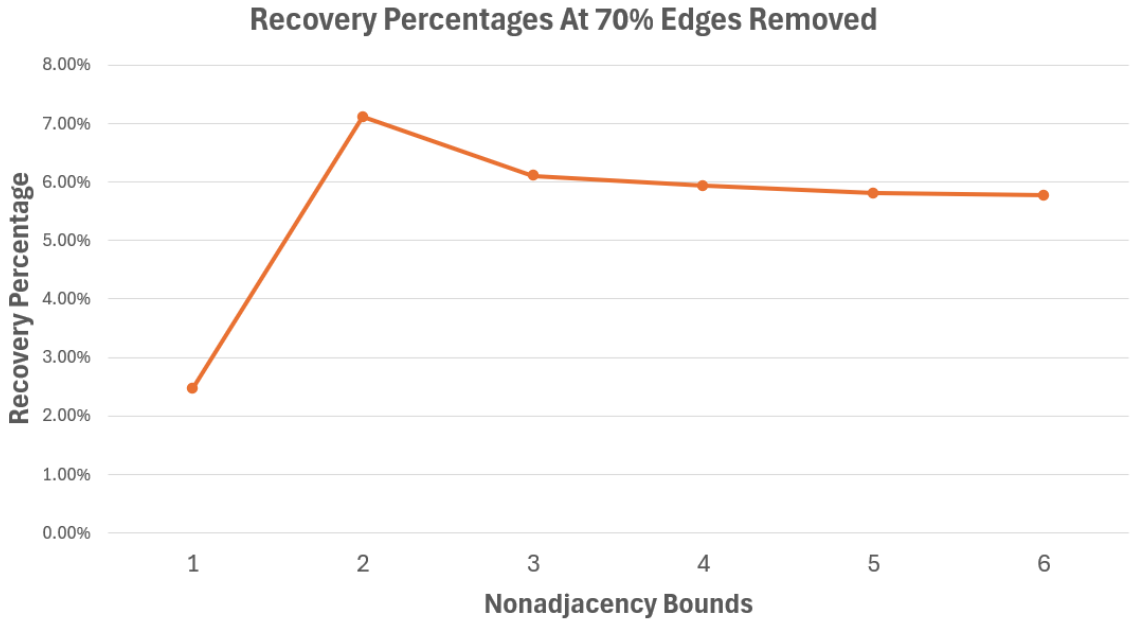
more indirect and uncertain measure of success, because little is likely to be known about a predicted edge unless it had also been removed.

We next applied our link prediction algorithm to each of these five tripartite graphs, varying the nonadjacency bound to study its effect. Consider, as a representative example, the graph focused on eye diseases with 70% of its edges deleted. In Figure 2.1, we show how the nonadjacency bound affects both the recovery rate and the recovery percentage for this graph. As another example, we scrutinized the five graphs with 1% of their edges removed. For these, we found that a nonadjacency bound of one averaged a recovery rate of barely over 35%, while a bound of two yielded an average rate just shy of 96%. This of course left little room for improvement, and larger bounds generated rates less than 99%. Results such as these suggest that a nonadjacency bound of two is superior to a bound of one, and furthermore that when reaching for bounds of three or more we experience greatly diminished returns and are thus unlikely to produce markedly better link predictions.

Drug repositioning is the main application of this effort, and so drug-protein link predictions are of course our foremost concern. Nevertheless, we shall now consider all three types of edge imputations.



(a) The edge recovery rates



(b) The edge recovery percentages

Figure 2.1: Typical edge recovery experience. We consider a tripartite graph whose partite sets represent diseases, drugs, and proteins. It was anchored by eye diseases, and 70% of its edges were pseudorandomly removed. (a) The recovery rate experienced a huge gain as the nonadjacency bound was raised from one to two, but then it developed a much more gradual trend as the bound was further increased. (b) Roughly in concordance, the recovery percentage reached its maximum value at a nonadjacency bound of two, and then it tapered off at larger nonadjacency values.

2.3 Evaluating Disease-Drug Link Predictions

Determining the quality of inferred relationships between diseases and drugs presents a special challenge because there is no underlying mechanistic foundation for such a relationship other than archival patterns of prescription. We therefore began by using the Tanimoto fingerprint similarity metric (72) to estimate the validity of disease-drug predictions. Our goal was to determine, given a disease-drug imputed edge, whether the drug was on average more structurally similar to known drugs prescribed for the same disease than it was to other drugs. We computed this metric using Rdkit (73) to obtain a drug’s molecular access system (MACCS) fingerprint (74), which is a 166-bit key drug structure descriptor widely used in molecular comparisons (75). We also used ccbmlib (76) to estimate p-values for the Tanimoto similarities. We did not conduct any sort of similarity comparison between diseases because each tripartite subgraph was constructed and analyzed separately based on diseases within the same major category. With this approach, we found support for the study of imputed edges in the form of modest but significant mean differential drug similarity. Details of the analysis are rather technical, however, and can be found in Appendix A.1.

We next looked at all edges that were missing from our original data in order to gauge how they compared to those we imputed. Let disease x and drug y denote a pair of non-neighbor vertices in one of our original five subgraphs. We identified all drugs adjacent to x in the relevant tripartite subgraph and formed the set $S_1(x, y)$, comprising Tanimoto similarity scores between these drugs and y . We also identified all drugs (excluding y) in the relevant tripartite subgraph that were not adjacent to x and built the set $S_2(x, y)$, using Tanimoto similarity scores between these drugs and y . We then ranked the missing disease-drug edges in each tripartite subgraph based on the difference between the means of these two sets of similarity scores. After this, we chose a cutoff value, n , selected the n highest ranked edges from each tripartite subgraph, and formed their respective unions S1 and S2. As before, we excluded edges containing diseases adjacent to fewer than three drugs since so little data was

available for them and to ensure an equitable comparison with triclique augmentation. We began with n set to 24, which is the smallest number of missing edges in any of the five tripartite subgraphs, and then varied n from 100 to 1000 in increments of 100. Across all these cutoff values, the proportion of highest ranked edges imputed by triclique augmentation consistently ranged from 3.4% to 5.0%. Thus, the overlap was low, providing additional support for the potential value added by the highly focused triclique approach over mere similarity ranking.

While these two analyses are decidedly positive, our main objective remains to exploit the wealth of data provided by all three partite sets. And thus we ask: how do proteins figure into disease-drug imputations? Therefore, in a third study, we counted the number of proteins adjacent to both vertices in the highest ranked disease-drug edges, comparing edges also imputed by triclique augmentation to all others. As shown in Figure 2.2, vertices incident to imputed edges shared more than eight common adjacent proteins, while vertices incident to non-imputed edges had just slightly over two. These results further strengthen the case that triclique augmentation effectively imputes disease-drug relationships for further study, and that by leveraging protein data it provides a possible mechanistic explanation for this stance.

2.4 Evaluating Disease-Protein Link Predictions

Estimating the potential significance of imputed associations between diseases and proteins was rather direct, largely because the interpretation of biological mechanisms underpinning human disease has long been a highly active area of research. To take advantage of prior work, we therefore employed two somewhat orthogonal methods, namely, Gene Ontology (GO) and Basic Local Alignment Search Tool (BLAST).

For GO, we concentrated on the molecular function enrichment of proteins that shared imputed edges with a common disease and conducted a literature review to assess the biological relevance of the most highly enriched GO terms. Our triclique

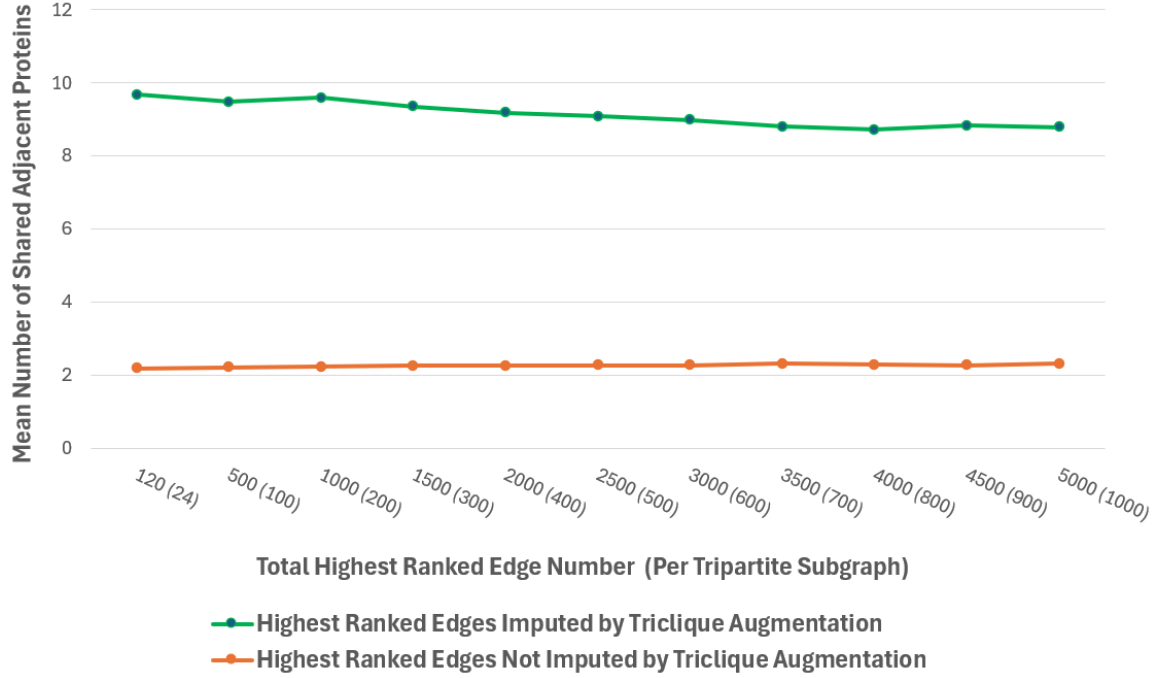


Figure 2.2: Mean number of shared adjacent proteins for diseases and drugs in highest ranked edges. The x-axis denotes the cutoff value, n , which limits the number of missing edges considered. The y-axis indicates the mean number of adjacent proteins shared between a disease and a drug in such an edge. The green line displays this number for the highest ranked missing edges imputed by triclique augmentation, while the orange line reflects the number for the highest ranked missing edges not imputed by triclique augmentation.

augmentation methodology imputed 2,578 disease-protein edges in the five graphs under study. These edges collectively covered 161 diseases and 199 proteins. From each tripartite subgraph, we chose the disease contained as an endpoint within the largest number of imputed disease-protein edges. The results of this selection were these: aortic valve insufficiency (61 edges), heat stroke (73 edges), macular degeneration (37 edges), mycoplasma infections (55 edges), and sarcoidosis (70 edges). For each of these five diseases, we next used ClueGO(77) to perform molecular function enrichment on the five respective sets of proteins adjacent to these diseases via imputed edges. It is important to note that ClueGO reports GO categories in groups, each group denoted by its most significant term. GO categories with Bonferroni-corrected p-values less than 0.01 were considered significantly enriched and are listed in Table 2.3.

We next conducted a literature review for some of the most significantly enriched GO categories to determine whether the corresponding disease associations were novel or supported by previous research. We focused our efforts primarily on GO categories that were enriched for two or more of the diseases under review. Let us discuss a few of these findings.

- G protein-coupled amine receptors were enriched in the analyses of aortic valve insufficiency and macular degeneration, while G protein-coupled serotonin receptors were enriched in analyses of heat stroke and sarcoidosis. Both receptor types belong to the G Protein-Coupled Receptor (GPCR) family, which forms the largest receptor family in the human genome, with endogenous ligands including odors, hormones, neurotransmitters, and chemokines (78). GPCRs are important in therapeutic applications because they are well-studied and serve as cell surface targets for various drugs (79). With this in mind, it is not surprising that they came up in our study. Moreover, newly predicted GPCR links such as these may provide novel insights into the treatment of disease.

Table 2.3: Molecular function enrichment by GO category and disease classification.

Enriched GO Category	Adjusted P-Value	Gene Coverage
Aortic Valve Insufficiency		
neurotransmitter receptor activity	1.81E-28	39.47%
organic acid binding	5.85E-15	25.00%
steroid hydroxylase activity	4.98E-09	15.79%
voltage-gated cation channel activity	2.30E-08	13.16%
G protein-coupled amine receptor activity	4.66E-05	6.58%
Heat Stroke		
neurotransmitter receptor activity	1.01E-30	27.88%
postsynaptic neurotransmitter receptor activity	3.44E-21	25.96%
steroid hydroxylase activity	4.33E-16	18.27%
G protein-coupled serotonin receptor activity	5.93E-11	11.54%
retinoic acid binding	8.20E-06	8.65%
oxidoreductase activity, acting on paired donors	3.45E-05	2.88%
nuclear receptor activity	1.53E-04	4.81%
Macular Degeneration		
neurotransmitter receptor activity	4.17E-36	28.40%
Continued on next page		

Table 2.3: Continued.

Enriched GO Category	Adjusted P-Value	Gene Coverage
G protein-coupled amine receptor activity	1.03E-28	18.52%
postsynaptic neurotransmitter receptor activity	3.27E-17	19.75%
transmitter-gated ion channel activity	3.06E-15	16.05%
adrenergic receptor activity	2.17E-11	6.17%
Gq/11-coupled serotonin receptor activity	5.37E-08	6.17%
oxidoreductase activity, acting on paired donors	1.13E-05	4.94%
Mycoplasma Infections		
neurotransmitter receptor activity	1.89E-26	33.87%
postsynaptic neurotransmitter receptor activity	1.49E-17	30.65%
glucuronosyltransferase activity	6.91E-10	14.52%
regulation of lipase activity	2.89E-06	9.68%
monooxygenase activity	3.97E-06	11.29%
Sarcoidosis		
neurotransmitter receptor activity	2.70E-34	26.72%
G protein-coupled serotonin receptor activity	2.31E-28	27.59%
gated channel activity	7.29E-16	18.97%
Continued on next page		

Table 2.3: Continued.

Enriched GO Category	Adjusted P-Value	Gene Coverage
postsynaptic neurotransmitter receptor activity	5.54E-14	10.34%
retinoic acid binding	2.83E-08	9.48%
monooxygenase activity	1.59E-06	6.90%

- Monooxygenase activity was enriched in the analyses of mycoplasma infections and sarcoidosis. Both diseases are associated with oxidative stress, with the former inducing it in host cells (80) and the latter characterized by increased levels(81). Certain monooxygenases, such as cytochrome P450 enzymes, also contribute to reactive oxygen species generation and modulate inflammatory responses. Biological processes of this nature may help explain why these proteins and diseases were linked by imputed edges.
- Neurotransmitter receptor activity was enriched in the analyses of all five diseases. These receptors play a crucial role in pain signal transmission and other signal transduction functions (82). Notably, this category provided the largest gene coverage in four of the analyses, except for sarcoidosis where it ranked second. Postsynaptic neurotransmitter receptor activity in particular was enriched for four of the diseases we considered, except for aortic valve insufficiency. Neurotransmitter receptors have previously been found to play a role in several relevant processes. Serotonin receptors, for example, are involved in heart valve tissue fibrosis (83), while ionotropic receptor expression may be enhanced by acute heat stress in aged rats(84). It is also known that glutamate, an excitatory neurotransmitter, serves as the primary driver of light responses in the retina (85), possibly linking it to macular degeneration. Additionally, some cases of mycoplasma pneumonia infection have been linked to recurrent anti-NMDA receptor encephalitis(86).
- Steroid hydroxylase activity was enriched in the analyses of aortic valve insufficiency and heat stroke. These enzymes play critical roles in the biosynthesis and metabolism of steroid hormones(87), such as testosterone, which can influence the heart’s structure and function (88). Previous studies in cows have demonstrated that heat stress too can affect the production and utilization of these hormones (89).

For BLAST, we used an NCBI tool [44] to assess protein similarity. We constructed sets $S_1(x, y)$ and $S_2(x, y)$ following the steps outlined in Section 2.3, using BLAST bit scores of protein sequences in the place of Tanimoto similarity scores of drug fingerprints. The BLAST bit score is a normalized measure of sequence similarity that represents the overall quality of an alignment between two sequences [45]. Additionally, for each BLAST alignment, we recorded the corresponding expected value (E-value). The E-value is a statistical measure that reflects the likelihood of observing the alignment by chance, with lower values indicating greater significance [46]. We followed the analysis steps outlined in Section 2.3 to study the number of shared adjacent drugs between diseases and proteins in imputed edges. As illustrated in Figure 2.3, vertices incident to imputed edges shared, on average, more than fourteen common adjacent drugs, whereas non-imputed edges shared, on average, fewer than one. These findings further support the effectiveness of triclique augmentation in utilizing information from all three partite sets for link prediction.

We also found support for the study of imputed edges in the form of significant mean differential protein similarity. Details can be found in Appendix A.2. At this point it is probably relevant to note that running BLAST on large numbers of proteins can require considerable computational time and resources. This further underscores the relative merit of triclique augmentation over merely ranking missing disease-protein edges based on vertex similarity.

2.5 Evaluating Drug-Protein Link Predictions

We identified a total of 42,323 unique imputed drug-protein edges across the five tripartite subgraphs under study. Establishing the feasibility of these imputed drug-protein associations by way of physical interactions was straightforward due to advancements in the molecular docking simulation software that we used to determine binding energies between drugs and proteins. Specifically, we calculated these energies using Autodock Vina (90), Molecular Operating Environment (MOE) (91), and

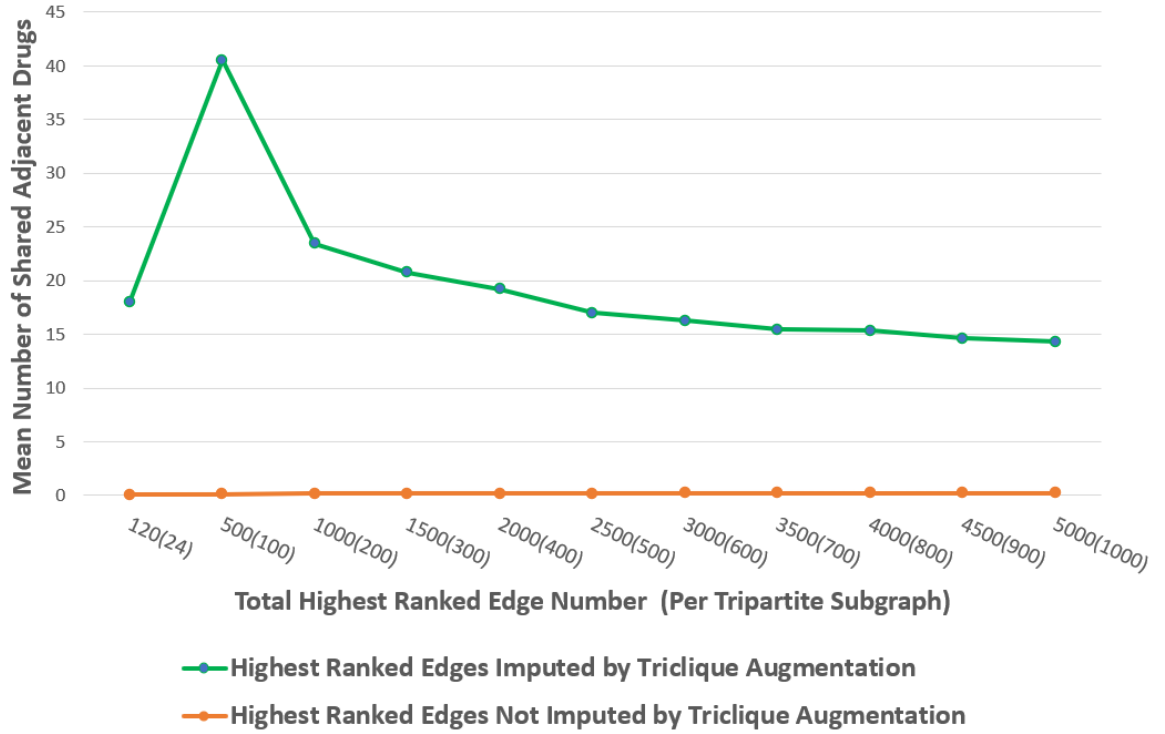


Figure 2.3: Mean number of shared adjacent drugs for diseases and proteins in highest ranked edges. The x -axis denotes the cutoff value, n , which limits the number of missing edges considered. The y -axis indicates the mean number of adjacent drugs shared between a disease and a protein in such an edge. The green line displays this number for the highest ranked missing edges imputed by triclique augmentation, while the orange line reflects the number for the highest ranked missing edges not imputed by triclique augmentation.

VinaMPI (92). We downloaded drug compounds from DrugBank in concatenated 3D-SDF format, processed and converted SDF files to Mol2 files using Open Babel (93), and transformed these into PDBQT (Protein Data Bank, Partial Charge and Atom Type) files using AutoDockTools (ADT) scripts (94). We pulled protein structures used in binding energy calculations from the RCSB Protein Data Bank (PDB) (95). Note that a single protein can exhibit multiple crystal structures in the PDB. Thus, to ensure optimal use of these structures, we gave preference to those with a higher number of amino acids and a higher resolution. We conducted a global search docking simulation using the entire protein without specifying a binding site. To accomplish this, we used VMD (96) to obtain the coordinates of a docking box that encapsulates the protein. Next, we converted the cleaned proteins into PDBQT files and used high-throughput virtual docking [49] to calculate the binding energies between drug ligands and protein receptors, retaining the binding energy of the best model for further analysis. Finally, we validated our docking protocol by comparing multiple docking results against known crystal structures (see Appendix A.3 for details).

As case studies, we selected five proteins (3TBG, 6KUW, 7C61, 7EJ8, and 7JVR) that we previously used in docking protocol validation and that satisfied our docking criteria to evaluate the predicted drug-protein interactions. For each of these proteins, we divided the drugs from the five tripartite subgraphs into three distinct sets: (1) drugs adjacent to the protein via imputed edges, (2) drugs adjacent to the protein via known edges, and (3) drugs not adjacent to the protein via either imputed or known edges. For protein p , we conducted molecular docking between p and drugs from these three sets and recorded their corresponding binding energies as $E_1(p)$, $E_2(p)$, and $E_3(p)$, respectively. Subsequently, we conducted Mann-Whitney U tests between $E_1(p)$ and $E_2(p)$, as well as between $E_1(p)$ and $E_3(p)$. For all five proteins, we observed that $E_1(p)$ exhibited lower binding energy values when compared to $E_3(p)$, with p -values less than 0.05 (Table 2.4). Additionally, for all five proteins, we found that the binding energy values in $E_1(p)$ showed no statistically significant difference when compared to $E_2(p)$. These results, alongside previous findings that lower binding

Table 2.4: Binding energy metrics and Mann-Whitney test results.

	PDB				
	3TBG	6KUW	7C61	7EJ8	7JVR
$\mathbf{E}_1(\mathbf{p})$ Mean	-7.59	-7.19	-6.76	-6.62	-6.74
$\mathbf{E}_2(\mathbf{p})$ Mean	-7.66	-7.12	-7.13	-6.62	-7.13
$\mathbf{E}_3(\mathbf{p})$ Mean	-7.09	-6.79	-6.36	-6.28	-6.41
$\mathbf{E}_1(\mathbf{p}) - \mathbf{E}_2(\mathbf{p})$ P-value	0.26	0.48	0.50	0.62	0.03
$\mathbf{E}_1(\mathbf{p}) - \mathbf{E}_2(\mathbf{p})$ Effect Size	0.06	-0.06	0.30	0.00	0.31
$\mathbf{E}_1(\mathbf{p}) - \mathbf{E}_3(\mathbf{p})$ P-value	7.80E-05	3.40E-05	3.00E-06	9.00E-05	6.73E-04
$\mathbf{E}_1(\mathbf{p}) - \mathbf{E}_3(\mathbf{p})$ Effect Size	-0.34	-0.29	-0.31	-0.27	-0.23

energy values indicate a higher probability of ligand-receptor interaction and complex formation [54], suggest that our algorithm did indeed select plausible drug-protein pairs.

2.6 Discussion

With this chapter we presented a novel, scalable, link prediction strategy for k -partite graphs, explicated it for $k = 3$ using a disease-drug-protein testbed we created from BioSNAP, and evaluated its potential utility at predicting all three types of links. Among its key analytic features are mathematical abstraction, which is provided through an application of graph theoretical algorithms, and interpretability, which is often missing with machine learning and other black-box methods. Notably, the procedures we crafted need no training data or negative samples. Moreover, they can be used when handling heterogeneous data, thus making them well-suited for studies that include data from multiple experiments.

We hasten to mention a few methodological limitations. At most a modest set of proteins was scrutinized for validation. And only in silico models were employed, since wet-lab technologies fall well outside the scope of this effort. Furthermore, the approach we devised quite naturally relies on data quality and thus may be of limited utility in the face of cold start scenarios (problems in which a newly designed drug is introduced with no previously known associations). We also note that docking may

not always correlate particularly well with experimental binding energy. Nevertheless, true binding compounds generally exhibit better energies than non-binding ones.

Chapter 3

Faster Maximal K -partite Clique Enumeration (MPCE)

Maximal k -partite cliques are widely used in various research fields, including social sciences, textile engineering, data mining, and particularly life sciences. However, enumerating maximal cliques or maximal k -partite cliques can be a time-consuming process. Prior research has proposed methods for improving the performance of maximal clique enumeration in general graphs. In the context of k -partite graphs, studies have focused on identifying k -cliques (i.e., cliques of size k) and computing maximum k -partite cliques. In contrast, work on enumerating maximal k -partite cliques remains limited. In this chapter, we present a faster maximal k -partite clique enumeration algorithm based on the Bron-Kerbosch algorithm with pivoting, which significantly speeds up the enumeration process. The performance of the algorithm is evaluated on both pseudorandomly generated graphs and real-world data. The source code for our algorithm is available on GitHub.

3.1 Introduction

A graph $G = \{V, E\}$ is a flexible data structure commonly used across various research domains, where V is a set of vertices representing instances in data (e.g., proteins), and E is a set of edges representing relationships between vertices (e.g., protein–protein interactions). The sizes of these sets are denoted by $|V|$ and $|E|$, respectively. All the graphs discussed in this chapter are simple, finite, and undirected. A clique is a complete subgraph where every pair of vertices is connected by an edge. A clique of k vertices is called a k -clique. A k -partite graph is a special structure in which vertices are divided into k independent partite sets with no edges within the same set, allowing researchers to focus on inter-partite relationships. In this context, k -partite cliques are complete subgraphs within a k -partite graph that include vertices from every partite set. A k -partite clique that cannot be extended by adding another vertex is called a maximal k -partite clique, and the largest of these maximal k -partite cliques is referred to as a maximum k -partite clique. Figure 3.1 shows an example of a k -partite graph, highlighting both a maximum and maximal k -partite clique.

K -partite cliques are widely applied in various research fields. In the social sciences, they are used to identify diverse communities (28), leveraging the ability of k -partite graphs to accommodate more than two groups. In textile engineering, they are applied in braiding processes (33), and in data mining, they aid in clustering categorical datasets (35). Their unique structure has also been used to solve specific scheduling problems, such as the job shop scheduling problem (97) and railway timetabling (36). Their usage is particularly prominent in life sciences, such as functional coherence analysis of proteins (23), heterogeneous functional genomics analysis (98), or indirect application in detecting and testing altered brain connectivity networks (25).

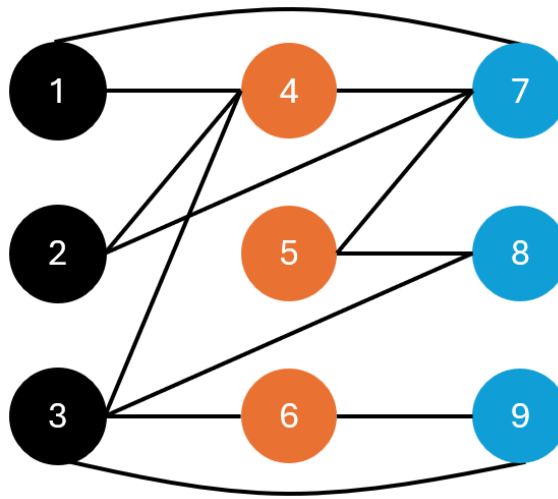


Figure 3.1: Example of a tripartite graph and its tripartite cliques. Vertices of the same color belong to the same partite set. The subgraphs $(1, 2, 4, 7)$ and $(3, 6, 9)$ are both maximal k -partite cliques; however, only the former is a maximum k -partite clique.

Before delving into k -partite clique enumeration, we first introduce the Bron–Kerbosch (BK) algorithm (42), which forms the foundation for several algorithms discussed in this paper. Originally proposed by Coen Bron and Joep Kerbosch in 1973, the BK algorithm is a widely recognized and efficient method for enumerating all maximal cliques in a graph. Enumerating all maximal cliques is computationally intensive, as the number of maximal cliques can grow exponentially with the size of the graph—specifically, up to $O(3^{|V|/3})$ (41). The BK algorithm uses a recursive backtracking approach systematically to explore all potential cliques in a graph. A particularly efficient variant incorporates a pivoting strategy, which minimizes redundant exploration by reducing the number of recursive calls. The pseudocode for this pivot-based version is presented in Algorithm 1. Moreover, the algorithm does not require storing cliques as they are found, thereby minimizing memory usage and improving scalability for large and complex graphs. Owing to its adaptability and performance, the Bron–Kerbosch algorithm remains a fundamental tool in combinatorial optimization and graph theory.

The enumeration of all maximal k -partite cliques has been less frequently explored, while there have been several studies focused on enumerating k -cliques and maximum k -partite cliques in partite graphs (97; 99). The maximal multipartite clique enumeration (MMCE) algorithm (41), published in 2019, is a notable exception. This algorithm first converts the k -partite graph G into a general graph G' by adding intra-partite edges and then applies the Bron–Kerbosch algorithm with pivoting to enumerate maximal cliques in G' . For each maximal clique found, MMCE checks whether it spans all partite sets and includes it in the output if it does.

One limitation of MMCE is that it does not identify whether a candidate partite clique is a k -partite clique until the candidate can no longer be extended. In other words, the algorithm spends time enumerating m -partite cliques ($2 \leq m < k$) in the graph, even though these are not stored in the output. Figure 3.2 displays examples of bipartite cliques in tripartite graphs. Our objective is to improve the maximal k -partite clique enumeration algorithm by eliminating candidates from the search tree

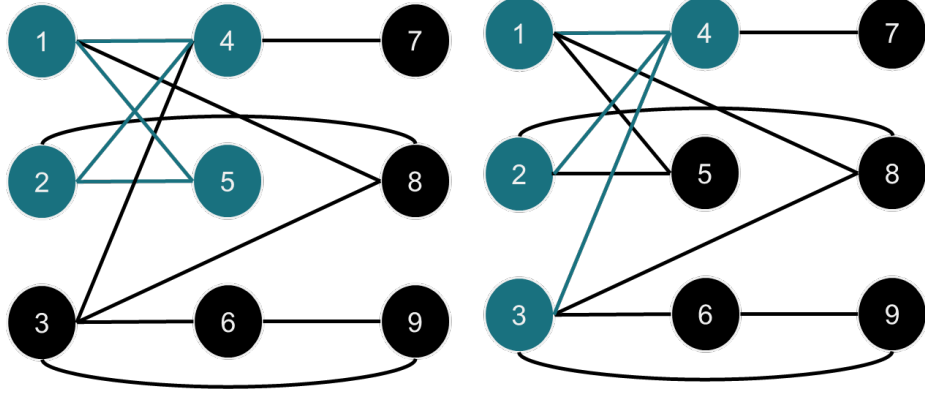


Figure 3.2: Example of two maximal bipartite cliques (green) in tripartite graphs

at an earlier stage if they cannot be extended to a k -partite clique that covers all partite sets, thereby reducing the practical running time.

In the following sections, we discuss the algorithm details in Section 3.2. In Section 3.3, we evaluate our algorithm’s performance using both artificially generated pseudo-random graphs and real-world datasets. In Section 3.4, we highlight the advantages of our algorithm and potential directions for future research.

3.2 Algorithm

As shown in Algorithm 1, the Bron–Kerbosch algorithm with pivoting recursively processes three sets: set C , which stores the vertices in the current candidate clique; set M , which contains vertices that may expand the current clique; and set X , which stores the vertices that cannot be used to expand the current clique. In each recursion, when a vertex v is added to C , the new candidate set is $M_{new} = M \cap N(v)$ and the new excluded set is $X_{new} = X \cap N(v)$, where $N(v)$ represents the adjacent neighbors of v .

To adapt the BK algorithm for k -partite clique enumeration, we implemented a straightforward modification: when adding v to C , we defined:

Algorithm 1 Bron–Kerbosch Algorithm with Pivoting (BK-Pivot)

Input: An undirected graph $G = (V, E)$; current clique C (initially $\emptyset$), candidate vertices M (initially V), and excluded vertices X (initially $\emptyset$)

Output: All maximal cliques in G

```
1: procedure BK-PIVOT( $C, M, X$ )
2:   if  $M = \emptyset$  and  $X = \emptyset$  then
3:     report  $C$ 
4:     return
5:   end if
6:   Choose a pivot vertex  $u$  from  $M \cup X$ 
7:    $M' \leftarrow M \setminus N(u)$ 
8:   for each vertex  $v$  in  $M'$  do
9:     BK-PIVOT( $C \cup \{v\}, M \cap N(v), X \cap N(v)$ )
10:     $M \leftarrow M \setminus \{v\}$ 
11:     $X \leftarrow X \cup \{v\}$ 
12:   end for
13: end procedure
```

Definitions:

C : Current clique being constructed

M : Set of vertices that can be added to extend C

$N(v)$: Set of neighbors of vertex v in graph G

X : Set of vertices that should be excluded to avoid duplicates

$$\begin{aligned}
M_{\text{new}} &= M \cap (N(v) \cup P(v)) \\
X_{\text{new}} &= X \cap (N(v) \cup P(v))
\end{aligned}
\tag{3.1}$$

where $P(v)$ denotes the vertices in the same partite set as v . This approach achieves the same effect as adding intra-partite edges, as done in the MMCE algorithm, but without the overhead of explicitly adding those edges, thus saving computation time. We refer to this approach as the MMCE variant. Our algorithm builds upon this variant, and we compare it with our own methods in terms of both space and time complexity in the following sections.

3.2.1 Pruning the Search Tree and Pseudo Code

Algorithm 2 presents the pseudocode for our enhanced Maximal k -Partite Clique Enumeration (MPCE) algorithm. Building upon and optimizing the MMCE variant, our algorithm follows the update framework described in Equation 3.1 to maintain the sets M and X , as reflected in lines 19–20 of Algorithm 2. Unlike the MMCE variant, our algorithm filters out partite cliques that cannot be extended into a k -partite clique at an early stage of the search process. This is achieved through the following strategies:

1. Pruning search branches where neither C nor M contains a vertex from a required partite set (lines 2–6).
2. Ensuring that the first k vertices added to C are selected from k distinct partite sets (lines 13–14).

In certain applications, it may be desirable to focus only on those k -partite cliques that contain at least a specified lower bound of vertices in each partite set. To support this, we introduce an additional parameter lb , representing the specified minimum number, which is used to filter k -partite cliques after they are identified (Algorithm 2, line 8).

Algorithm 2 Maximal K -partite Clique Enumeration (MPCE)

Input: A k -partite graph $G = \{V, E\}$, the candidate partite clique C (initial with $\emptyset$), the candidate vertices M (initial with V), and the temporarily excluded vertices X (initial with $\emptyset$)

Output: All maximal k -partite cliques in G containing at least lb vertices in each partite set

```
1: procedure MPCE( $C, M, X$ )
  #Terminate if there exists a partite set such that neither  $C$  nor  $M$  contains a
  vertex from it.
2:   for  $i \leftarrow 0$  to  $k - 1$  do
3:     if  $(C \cap \text{partite\_sets}[i]) = \emptyset$  and  $(M \cap \text{partite\_sets}[i]) = \emptyset$  then
4:       return
5:     end if
6:   end for
  #Report  $C$  as a maximal  $k$ -partite clique
7:   if  $M = \emptyset$  and  $X = \emptyset$  then
8:     if  $|C \cap \text{partite\_sets}[i]| \geq lb$  for all  $i \in [0, k - 1]$  then
9:       report  $C$ 
10:      return
11:    end if
12:  end if
  #The first  $k$  vertices of  $C$  should be selected from  $k$  different partite sets.
13:  if  $\text{len}(R) < k$  then
14:     $M' = \{v \in M \mid v \text{ is in a partite set disjoint from } C\}$ 
15:  else
16:    Choose the first node  $u$  in  $M$  as the pivot
17:     $M' = M \setminus (N(u) \cup P(u))$ 
18:  end if
19:  for each vertex  $v$  in  $M'$  do
20:    MPCE( $R \cup \{v\}, M' \cap (N(v) \cup P(v)), X \cap (N(v) \cup P(v))$ )
21:     $M' \leftarrow M' \setminus \{v\}$ 
22:     $X \leftarrow X \cup \{v\}$ 
23:  end for
24: end procedure
```

Definitions:

C : The current partite clique

k : Number of partite sets

lb : Min. vertices per partite set in the returned k -partite clique

M : List of vertices that can potentially be added to the current partite clique

$N(u)$: Neighboring vertices of vertex u

$P(u)$: Vertices in the same partite set as u , excluding u

$\text{partite_sets}[i]$: Vertices belonging to partite set i

X : List of vertices excluded from the current partite clique and all its supersets

3.2.2 Time and Space Complexity

Observation 3.1. *The MPCE algorithm does not increase the time complexity compared to the MMCE variant.*

In each recursive call, our algorithm includes two enhancements compared to the MMCE variant mentioned in Section 3.2. The first enhancement is to check whether any partite set lacks vertices in both the candidate set (M) and the candidate clique (C). (Algorithm 2 lines 2-6). In the implementation, we used two arrays to track $|C_i|$ and $|M_i|$, which represent the number of vertices in the intersection between partite set i and the sets C and M , respectively. This check in each recursive call has a time complexity of $O(p)$, where p is the number of partite sets. In practice, p is typically much smaller than the vertex number $|V|$, so this overhead is generally negligible ($O(p) \sim O(1)$).

The second enhancement ensures that the first k vertices in the candidate clique are selected from k different partite sets (Algorithm 2, lines 13–14). To achieve this, when the recursion depth does not exceed k , a vertex in M_i is skipped if the size of C_i is non-zero. Since two arrays are used to track the sizes of M_i and C_i , each check has a time complexity of $O(1)$, and constructing M' in this step takes $O(|V|)$ time. When the recursion depth exceeds k , MPCE and the MMCE variant construct M' in the same way (lines 16–17 of the algorithm), which also requires $O(|V|)$ time in each recursive call. In both cases, the resulting M' , which includes the vertices to be processed in future iterations and recursive calls, is a subset of M with size $O(|V|)$. Therefore, MPCE does not increase the order of magnitude of the iteration and recursive calls, nor does it increase the time complexity in each recursive step. Both the MPCE and MMCE variant are built upon the BK algorithm. In the worst case, it can reach $O(3^{|V|/3})$, which matches the maximum possible number of maximal k -partite cliques (41). However, it is worth noting that, in real-world scenarios, the worst case rarely occurs (40).

Observation 3.2. *Determining whether a k -partite maximal clique contains at least lower bound (lb) vertices in each partite set can be achieved without increasing the overall time complexity of the recursive subroutine.*

When the candidate partite clique can no longer be extended, MPCE checks whether $|C_i| > lb$ for each i , which requires $O(p) \sim O(1)$ time since $p \ll |V|$. This filtering step is applied post hoc and does not affect the algorithm’s overall runtime.

Observation 3.3. *The MPCE algorithm does not increase the space complexity compared to the MMCE variant.*

In both the MPCE and the MMCE variant, the graph’s vertex and edge information must be stored, resulting in a space complexity of $O(|V| + |E|)$. Additionally, finding a maximal k -partite clique requires tracking the corresponding partite set of each vertex, adding an extra $O(|V|)$ space overhead. Both MPCE and MMCE are based on the BK algorithm and therefore maintain three sets of vertices (C , M , and X) in each recursive call. Managing these sets incurs a space complexity of $O(|V|)$ per call, leading to an overall space complexity of $O(|V| \cdot d)$, where d is the recursion depth. Unlike the MMCE variant, the MPCE algorithm also tracks the number of vertices included from each partite set in the current candidate clique. This requires an additional $O(p) \sim O(1)$ space per recursive call, where p is the number of partite sets. Therefore, the MPCE algorithm does not increase the space complexity per recursive call compared to the MMCE variant. As mentioned in Observation 3.1, since the recursion depth does not increase, the overall space complexity does not increase either.

3.3 Algorithm Performance Assessment

Both the MPCE algorithm and the MMCE variant were implemented in C, and the code was based on Yun Zhang’s implementation of the BK algorithm. The source code for MPCE can be found on GitHub <https://github.com/Cheng111/MPCE>. All

experiments were conducted on the ISAAC Next Generation (ISAAC-NG) cluster at the University of Tennessee, Knoxville. Each individual test was subject to a timeout limit of 5 hours.

We evaluated the algorithm on both pseudorandomly generated graphs and real-world data. For the artificially generated graphs, we controlled each test to vary only one parameter, allowing for a sensitivity analysis of the algorithm with respect to that specific variable. The real-world data provided a more robust assessment of the MPCE algorithm’s performance in practical scenarios.

3.3.1 Algorithm Evaluation on Pseudorandomly Generated Graphs

We generated pseudorandom graphs with varying parameters, including the number of partite sets, the number of vertices in each partite set, and the density. For each parameter setting, 10 test files were generated, and the results were presented using the mean running time and memory usage of these 10 files. The counts of k -partite cliques were also recorded to provide a better understanding of the running time.

Impact of Vertex Number on Performance

The first test we conducted using pseudorandomly generated graphs aimed to examine how the algorithm’s running time varied with the number of vertices. We applied two approaches to increasing the number of vertices: increasing the number of vertices in each partite set and increasing the number of partite sets. The detailed graph metric settings for each scenario were:

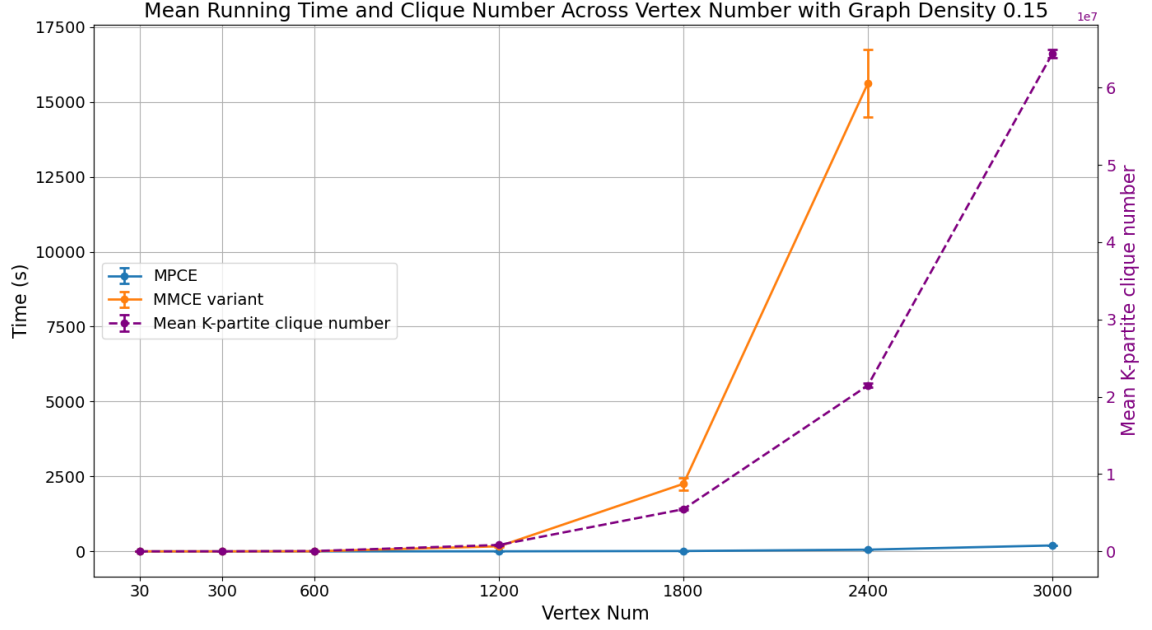
1. All the tested graphs were tripartite, with the total number of vertices $|V|$ set to one of the following values: 30, 300, 600, 1200, 1800, 2400, or 3000, and each partite set containing an equal number of vertices. Edges between partite sets were added with uniform density. We evaluated progressively increasing densities: 0.01, 0.02, 0.05, 0.1, 0.15, 0.2, and 0.25. For each fixed density, the

trends in running time and memory usage remained consistent across vertex counts. Therefore, only the results for density 0.15 are presented for further discussion (Figure 3.3), as it offers a good balance—it is dense enough to highlight performance differences between algorithms, while most tests at this level completed within the allotted time.

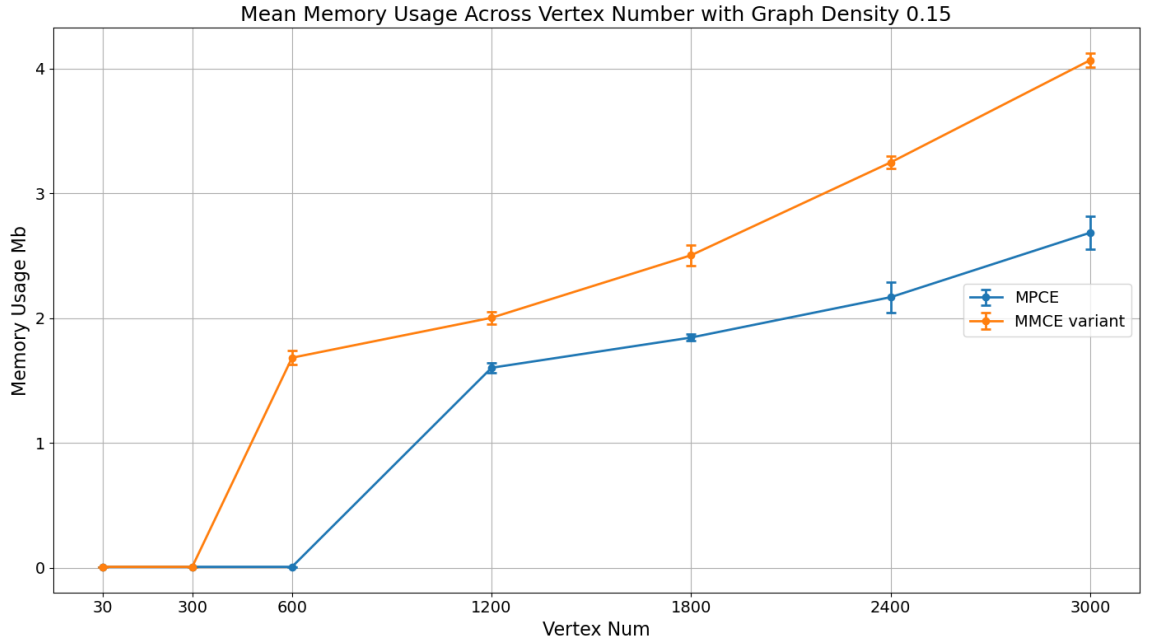
2. We tested the algorithms on k -partite graphs where each partite set contained 400 vertices, with a graph density of 0.15. The number of partite sets was incrementally increased from three to six.

In Figure 3.3a, we observed that, as expected, the running time increased with the number of vertices. Notably, as the graph became denser and the overall running time increased, the advantage of our algorithm became more evident—MPCE exhibited a slower growth rate in running time compared to the MMCE variant. The difference in memory usage between the MPCE and MMCE variant is not as pronounced as the difference in running time. However, as shown in the Figure 3.3b, MPCE consistently uses less memory, and the gap tends to widen as the number of vertices exceeds 1,200 and continues to grow.

Figures 3.4a and 3.4b showed how the running time changed with an increasing number of partite sets for the MPCE and the MMCE variant. These figures are separated in order to illustrate the trends better, as the difference in running times is substantial. In the MPCE tests, as the partite sets increased, we observed a decrease in running time when the number of partite sets increased from three to four, followed by an increase as the number continued to rise. In contrast, for the MMCE variant, the running time consistently increased as the number of partite sets grew from three to six. The number of cliques also steadily decreased with the increasing number of partite sets. Consistent with earlier observations, our MPCE algorithm exhibited a smoother and slower growth in running time compared to the MMCE variant.

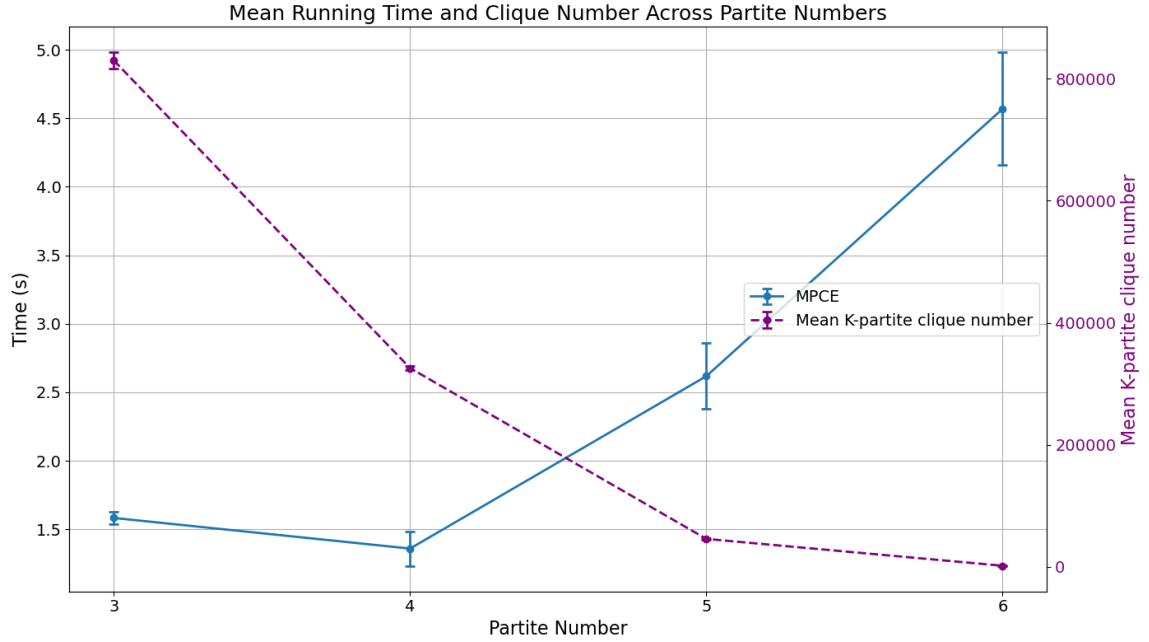


(a) Running time and the tripartite clique number change across vertex number

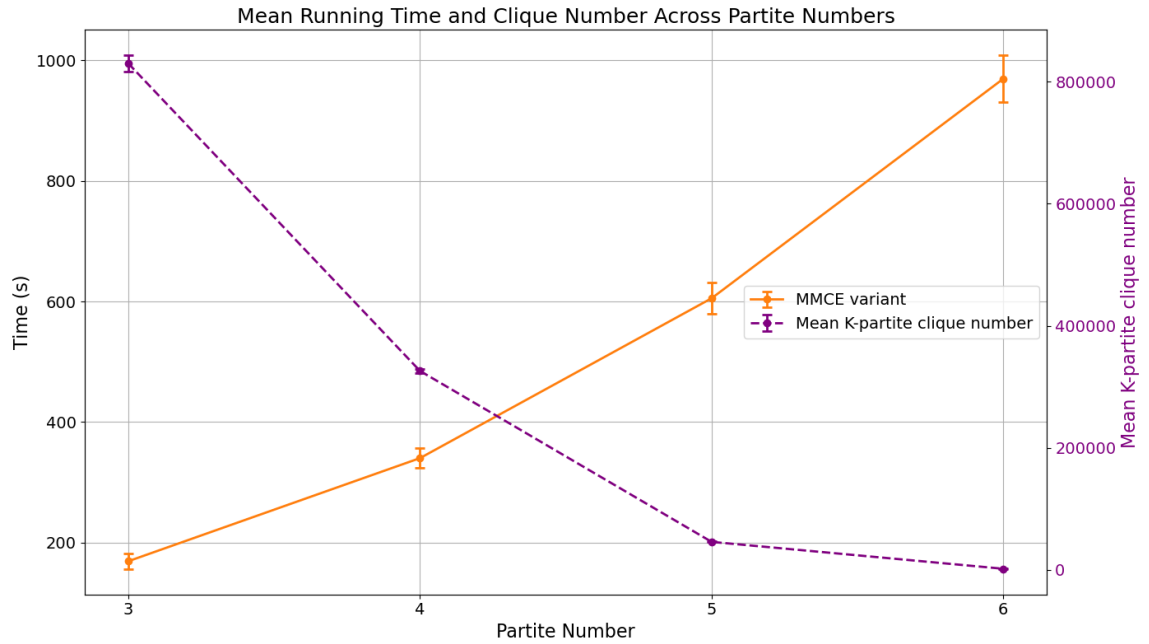


(b) Memory change across vertex number

Figure 3.3: How the running time, the number of tripartite cliques, and the memory usage changed as the number of vertices in each partite set increases. The number of vertices in each tripartite graph increases from 30 to 3000, with a uniform edge density of 0.15. The yellow line represents the MMCE variant, and the blue line represents MPCE, both corresponding to the left y-axis. The purple dashed line represents the number of k -partite cliques, corresponding to the right y-axis.

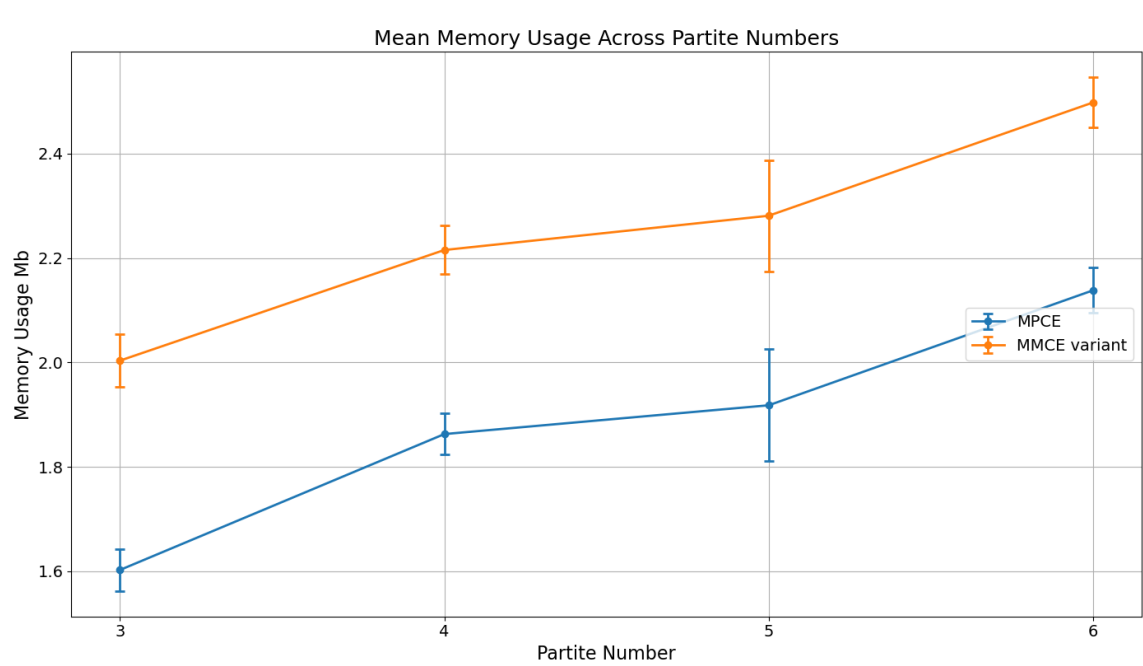


(a) MPCE running time and k -partite clique number versus the number of partite sets.



(b) MMCE variant running time and k -partite clique number versus the number of partite sets.

Figure 3.4: How the running time, the number of k -partite cliques, and the memory usage changed as the number of partite sets increases. The number of partite sets varies from three to six, with each set containing 400 vertices. All graphs have a uniform edge density of 0.15. The yellow line represents the MMCE variant, and the blue line represents MPCE, both corresponding to the left y-axis. The purple dashed line represents the number of k -partite cliques, corresponding to the right y-axis.



(c) Memory usage change across partite set number.

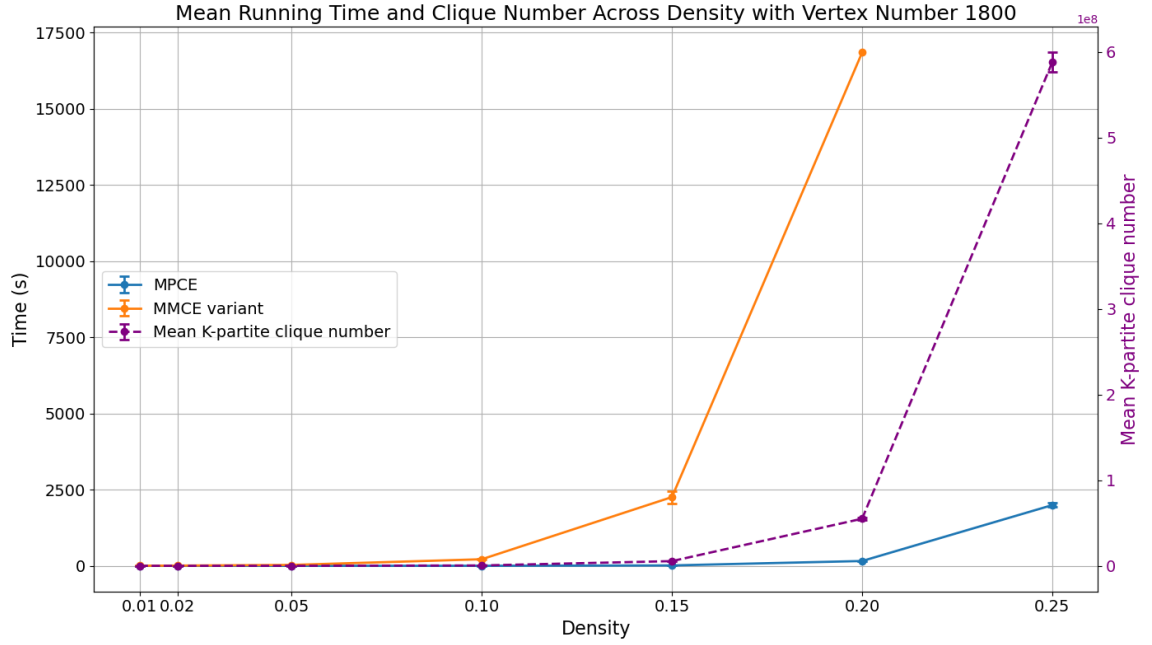
Figure 3.4: continued.

Notably, for six-partite graphs, the MMCE variant took around 1,000 seconds on average per test task, while the MPCE required less than 5 seconds per task on average. Additionally, as shown in Figure 3.4c, MPCE consistently used less memory than the MMCE variant.

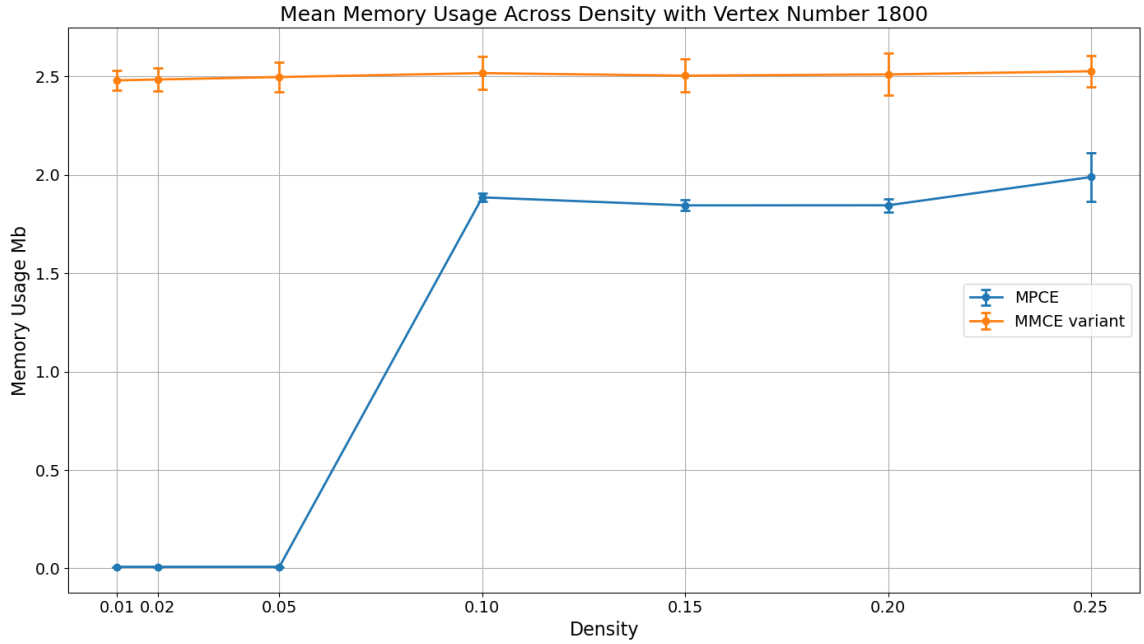
Impact of Density on Performance

To evaluate the algorithm’s performance across varying graph densities and to isolate the effect of density, we conducted experiments on tripartite graphs, ensuring that each partite set contained an equal number of vertices. Graph density was varied across the following values: 0.01, 0.02, 0.05, 0.10, 0.15, 0.20, and 0.25. We ran multiple experiments with total vertex counts ranging from 600 to 3000, in increments of 600. The results demonstrated consistent trends in both runtime and memory usage across all vertex configurations. Accordingly, we presented the results for the case with 1800 vertices—the median vertex count—as a representative example.

As the graph density increased, the running times of both the MMCE variant and MPCE also increased. However, the runtime of MPCE grew significantly more slowly than that of the MMCE variant (Figure 3.5a). Notably, when the density reached 0.25, the MMCE variant failed to complete the task within the allotted time, whereas MPCE completed it in under 2500 seconds on average. With the vertex count fixed at 1800, MMCE variant’s memory usage showed little variation across different graph densities (Figure 3.5b). A possible explanation is that, although the MMCE variant does not explicitly add intra-partite edges—as discussed in Section 3.2—it maintains the same C , M , and X sets and performs the same number of recursive calls as the original MMCE algorithm. Effectively, the MMCE variant behaves as if all intra-partite edges are present, resulting in a significantly larger effective edge set than that of the original graph. As a result, the algorithm processes a similar number of edges regardless of the initial graph density. The memory usage of MPCE algorithm increased with graph density but gradually leveled off, and it consistently consumed less memory than the MMCE variant across all tested densities.



(a) Running time and the number of tripartite cliques vary with density when $|V| = 1800$



(b) Memory usage varies with density when $|V| = 1800$

Figure 3.5: How the running time, the number of tripartite cliques, and the memory usage changed with the density. Each tripartite graph has an equal number of vertices in each part, totaling 1800 vertices. All graphs have a uniform edge density, which varies from 0.01 to 0.25 across the test cases. The yellow line represents the MMCE variant, and the blue line represents MPCE, both corresponding to the left y-axis. The purple dashed line represents the number of k -partite cliques, corresponding to the right y-axis.

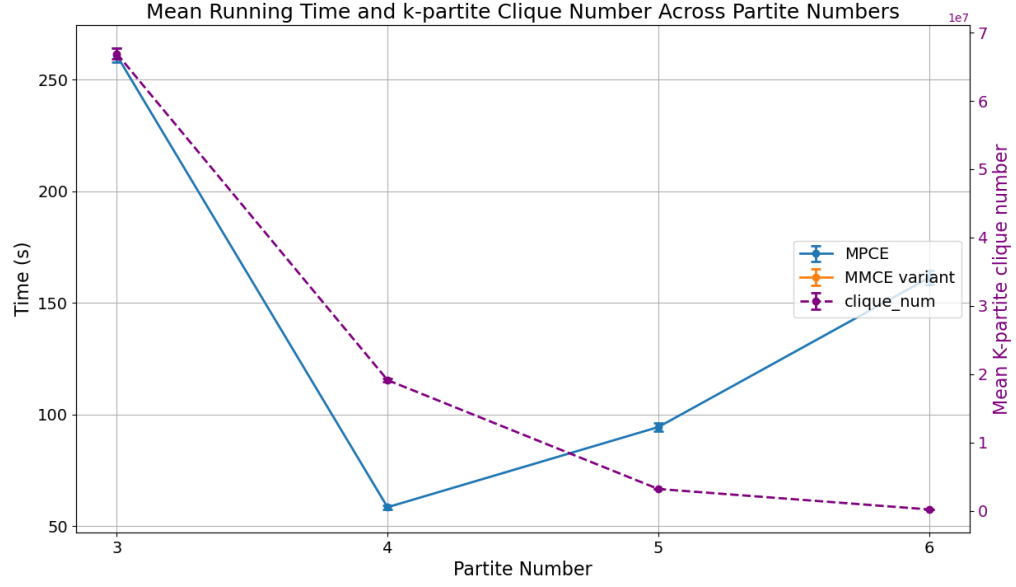
Table 3.1: Metrics of imbalanced test graphs

	Test Scenarios			
p_1 vertex number	1600	400	800	800
p_i vertex number ($i \neq 1$)	800	800	800	800
$p_1 - p_2$ inter-partite edge density	0.15	0.15	0.3	0.075
Other inter-partite edge density	0.15	0.15	0.15	0.15

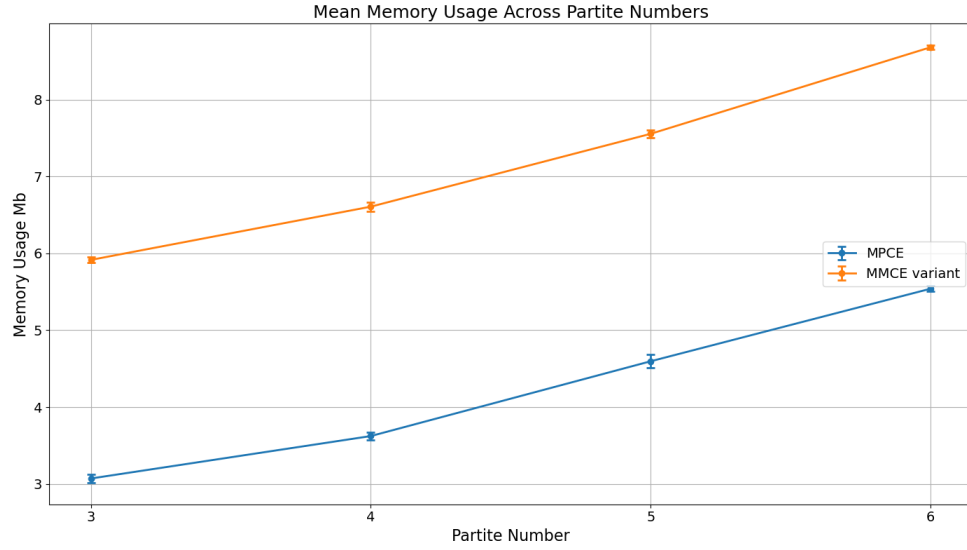
Performance on Graphs with Imbalanced Edge Counts

In previous sections, we analyzed the algorithms’ performance on graphs with evenly distributed vertex numbers and densities across the partite sets—an idealized scenario that rarely occurs in real-world applications. In this section, we shift our focus to k -partite graphs with imbalanced vertex distributions or edge densities. Specifically, we conducted experiments under four scenarios: (1) one partite set had twice as many vertices as the others; (2) one partite set had half as many vertices as the others; (3) one inter-partite edge set had twice the density of the others; and (4) one inter-partite edge set had half the density of the others. These tests were conducted on graphs ranging from tripartite to six-partite. The metrics of the test graphs for each scenario are summarized in Table 3.1, where p_i denotes the i -th partite set in the k -partite graph.

When one partite set contained more vertices than the others (Figure 3.6a), the MMCE variant timed out in all 40 test cases, whereas the MPCE algorithm successfully completed them all. Notably, as the number of partite sets increased from three to four, the running time of MPCE initially decreased but then began to rise again, despite a continued decline in the number of k -partite cliques. Moreover, MPCE consistently used less memory than the MMCE variant across all test cases (Figure 3.6b), further demonstrating its superior performance. In cases where one partite set contained fewer vertices than the others, we plotted the running times of the MMCE variant and MPCE in separate figures to provide a clearer view of the performance trends.

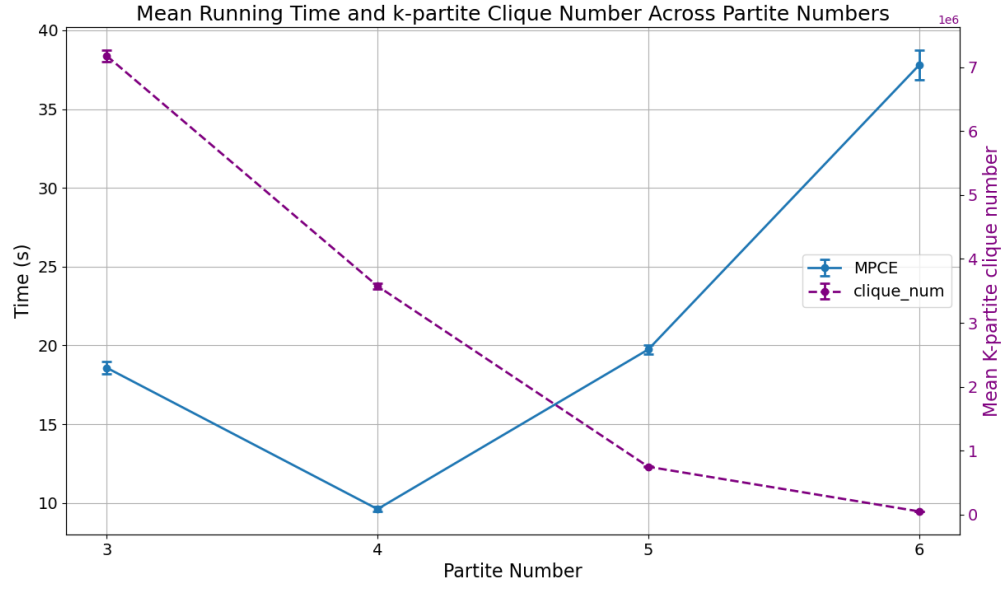


(a) Running time of MPCE when one partite set has 1600 vertices and the others have 800 vertices.

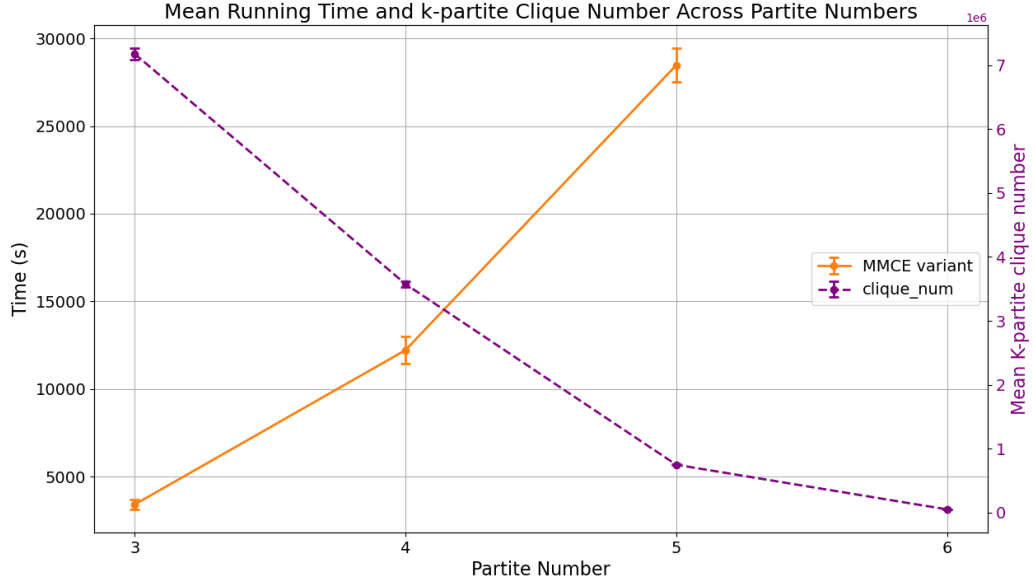


(b) Memory usage when one partite set has 1600 vertices and the others have 800 vertices.

Figure 3.6: Performance analysis on imbalanced k -partite graphs with uneven partite set sizes. The yellow line represents the MMCE variant, and the blue line represents MPCE, both corresponding to the left y-axis. The purple dashed line represents the number of k -partite cliques, corresponding to the right y-axis.

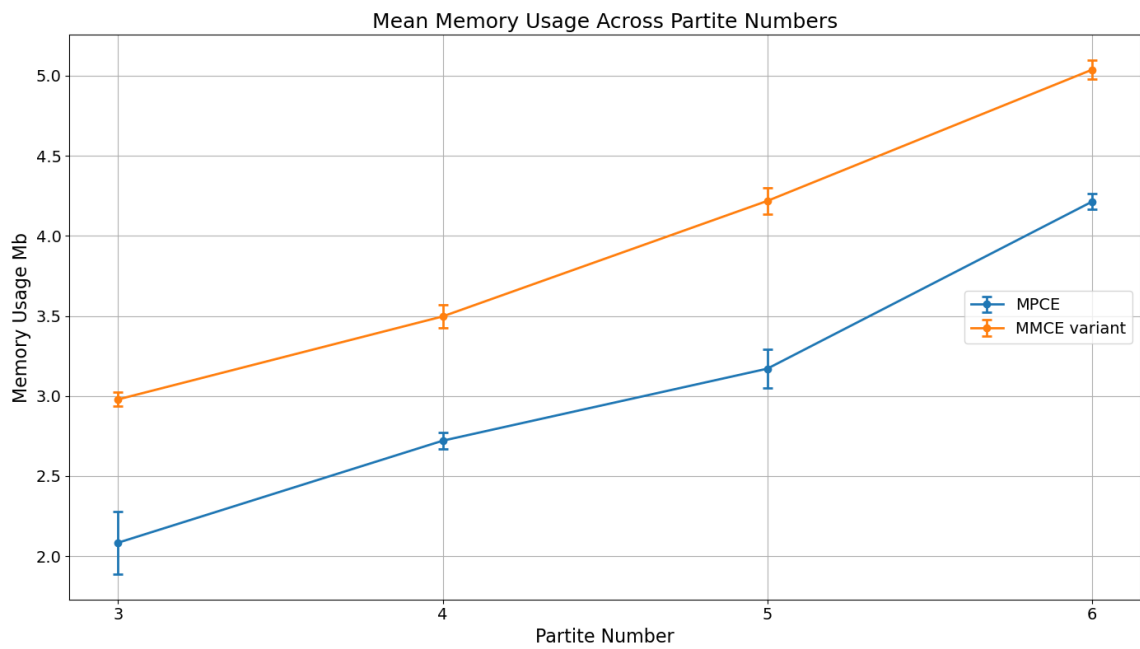


(c) Running time of MPCE when one partite's vertex number is 400 and others are 800.



(d) Running time of MMCE variant when one partite's vertex number is 400 and others are 800.

Figure 3.6: continued.



(e) Memory usage when one partite's vertex number is 400 and others are 800.

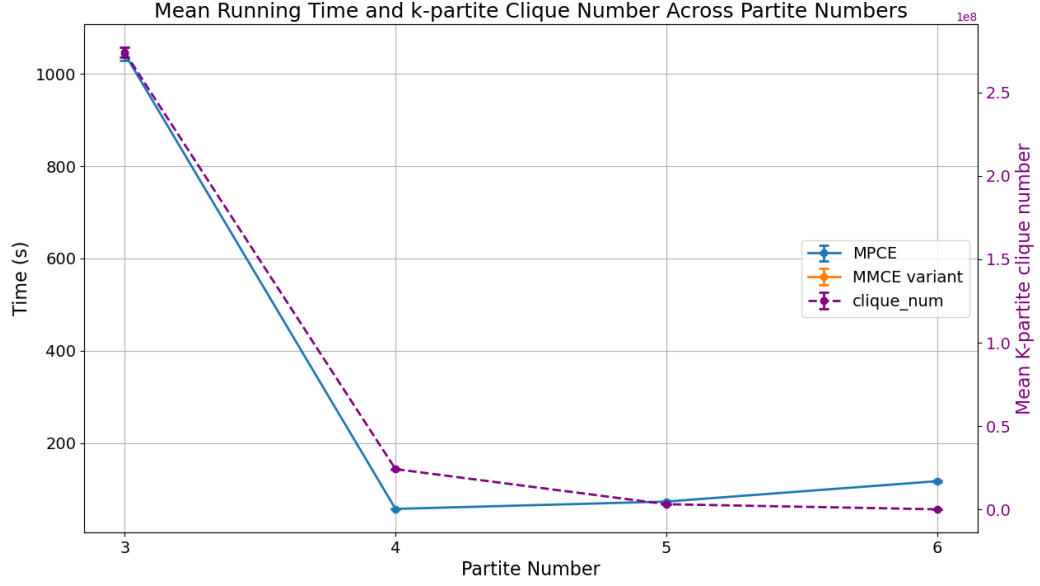
Figure 3.6: continued.

The MMCE variant required significantly more time—hundreds of times longer—than MPCE. Its running time increased as the number of partite sets grew and resulted in timeouts for all cases when the number of partite sets reached six (Figure 3.6d). In contrast, the running time of MPCE decreased when the number of partite sets increased from three to four, and then increased as the partite number continued to rise (Figure 3.6c). In this scenario, MPCE still consistently used less memory than the MMCE variant across all test cases (Figure 3.6e), with both methods exhibiting a similar trend as the number of partite sets increased.

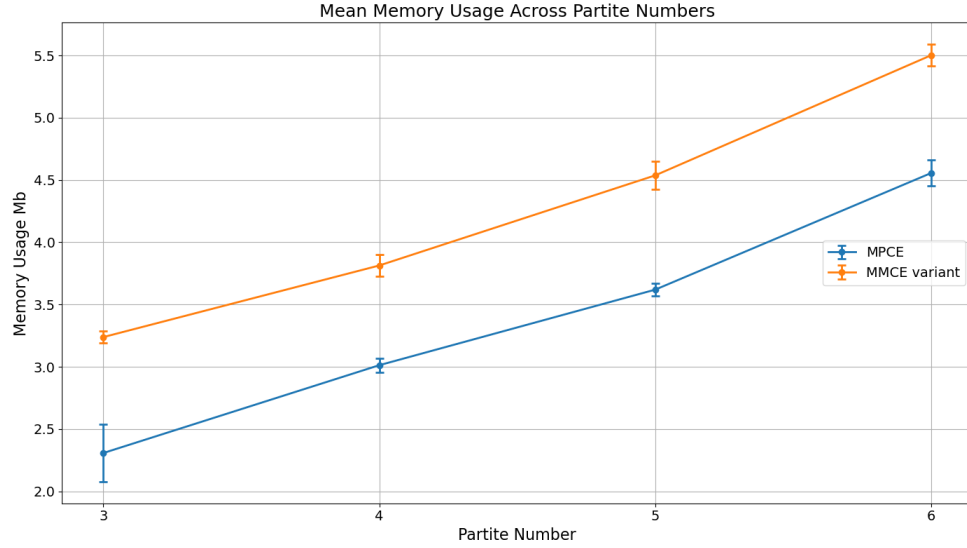
When one inter-partite edge set had twice the density of the others, the MMCE variant timed out on all 40 test cases. The running time of MPCE (Figure 3.7a) decreased as the number of partite sets increased from three to four, then gradually increased as the partite number continued to grow. When one inter-partite edge set had half the density of the others, the MMCE variant completed only the tripartite graph tests within the required time, timing out on all others. In contrast, MPCE successfully completed all tests, with its running time increasing as the number of partite sets grew (Figure 3.7c). In both scenarios, MPCE consistently used less memory than the MMCE variant (Figures 3.7b and 3.7d).

3.3.2 Algorithm Evaluation on Real-World Data

Given the widespread use of maximal cliques in the life sciences, we focused the algorithm evaluation on real-world data, specifically using datasets from the life sciences. The data sets we tested were from Stanford Biomedical Network Dataset Collection (BioSNAP) (64). BioSNAP provides relationships and interaction related to biological entities such as cells and drugs. We used three association lists from the collection: ChG-Miner (drugs-gene products), DCh-Miner (diseases-drugs), and DG-Miner (disease-gene products). Our focus was on diseases belonging to five categories: eye diseases, heart diseases, hemic and lymphatic conditions, infections, and wounds and injuries—each representing diverse affected tissues. Using these data,

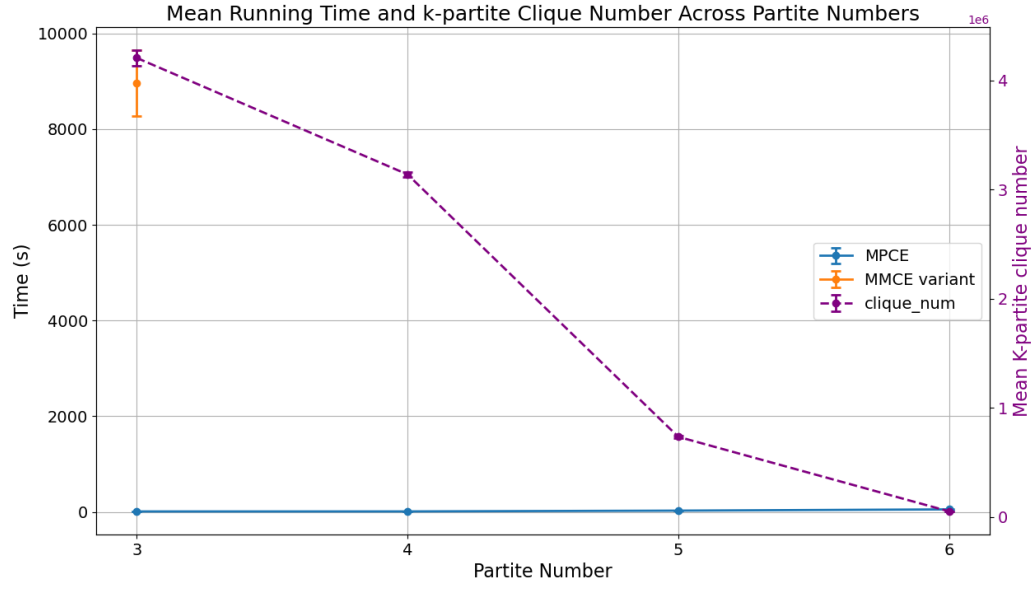


(a) Running time when one inter-partite edge set has a density of 0.3, while the others have a density of 0.15.

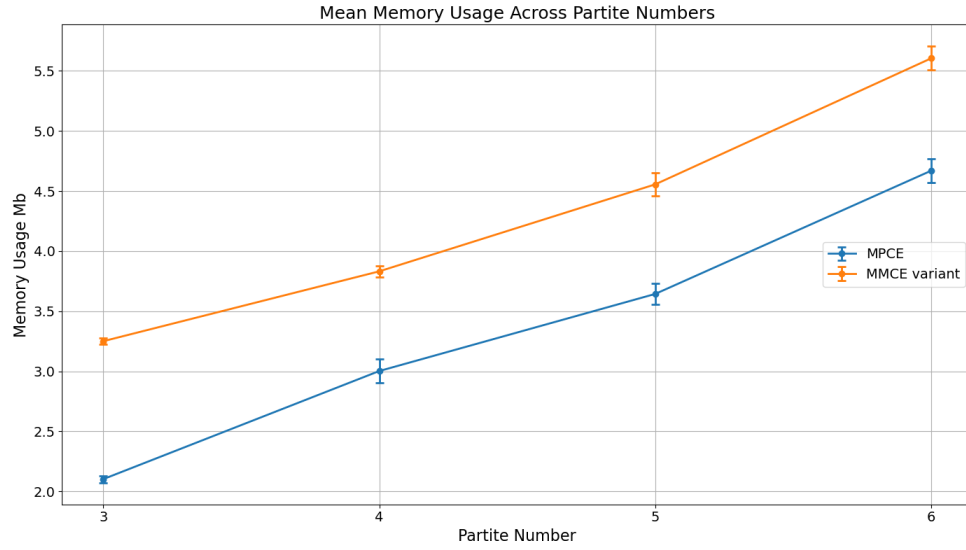


(b) Memory usage when one inter-partite edge set has a density of 0.3, while the others have a density of 0.15.

Figure 3.7: Performance analysis on imbalanced k -partite graphs with uneven inter-partite edge densities. The yellow line represents the MMCE variant, and the blue line represents MPCE, both corresponding to the left y-axis. The purple dashed line represents the number of k -partite cliques, corresponding to the right y-axis.



(c) Running time when one inter-partite edge set has a density of 0.075, while the others have a density of 0.15.



(d) Memory usage when one inter-partite edge set has a density of 0.075, while the others have a density of 0.15.

Figure 3.7: continued.

Table 3.2: Running time and data metrics on BioSNAP tripartite graphs

Graph Metrics	Tripartite Graphs				
	Eye Disease	Heart Disease	Hemic & Lymphatic	Infections	Wounds & Injuries
$ Disease $	88	40	79	150	71
$ Drug $	844	963	925	814	885
$ Protein $	1987	1992	1991	1977	1986
$ Disease - Drug $	5539	6630	9272	23068	6090
$ Disease - Protein $	59821	33524	74530	115283	31225
$ Drug - Protein $	5402	5826	5673	6232	5656
Density	0.04	0.02	0.04	0.07	0.02
Tripartite cliques	64826	43378	486981	60776	32532
Memory (MMCE variant) (MB)	7.06	7.21	7.13	7.18	7.05
Memory (MPCE) (MB)	2.64	3.03	3.00	3.13	3.15
Time (MMCE variant) (h)	> 24	> 24	> 24	> 24	6.07
Time (MPCE) (s)	8.53	11.04	18.69	6.37	4.31

we constructed five disease-drug-protein tripartite graphs by including disease vertices from the same category along with their adjacent drug and gene product vertices.

Compared to artificially generated graphs, we cannot control individual variables in graphs derived from real-world data, which means that differences in final running times may be influenced by multiple factors. In all five tripartite graphs, MPCE demonstrated significantly better performance than the MMCE variant. Table 3.2 presents the metrics of each tripartite graph and the performance of both the MMCE variant and MPCE. The first three rows show the number of vertices in each partite set, followed by three rows indicating the number of inter-partite edges between each pair of partite sets. Row seven includes the overall graph density, and row eight shows the number of triclques in each tripartite graph. The next two rows report the memory usage of the two algorithms, and the final two rows display their respective running times.

3.4 Discussion

The results of this study have significant implications for both theoretical and practical applications of graph theory. This faster maximal k -partite clique enumeration algorithm, MPCE, demonstrated superior performance—particularly in handling

large and dense graphs. This suggests that the algorithm is well-suited for real-world applications, where the complexity and size of data often pose challenges for traditional enumeration methods.

Furthermore, although we tested the algorithm only on real-world data from the life sciences, its performance demonstrates both robustness and scalability, making it a viable option for large-scale data analysis tasks in fields such as social network analysis, where identifying tightly connected subgroups is essential. The relatively slow increase in running time—even as graph density grows—underscores the algorithm’s potential for analyzing dense networks in applications such as telecommunications or transportation systems.

Chapter 4

The Power Gabriel Graph

Proximity graphs are fundamental data structures in computational geometry, designed to capture the spatial relationships between points. The Power Gabriel Graph (PGG) is introduced as a novel graph construction method inspired by a classic proximity graph—Gabriel Graph—and designed to enhance data-driven graph construction. PGG extends the traditional Gabriel Graph by incorporating a tunable power parameter, enabling flexible control over edge filtering strictness and overall graph density. The method is applied to gene expression datasets—an area where proximity graphs have seen limited use—and demonstrates the ability to produce more informative networks. We compared this algorithm with a widely used traditional gene graph construction method: spectral graph theory-based thresholding. Additionally, we combined the two approaches to leverage the strengths of both local and global methods. Empirical evidence indicates that applying higher powers leads to improved biological relevance in vertex clustering tasks. Notably, the algorithm preserves key theoretical properties, including the absence of isolated vertices and monotonic edge removal as power increases. These findings suggest that the Power Gabriel Graph provides an effective, data-driven approach for constructing biologically meaningful networks, with potential benefits for downstream analytical tasks.

4.1 Introduction

Graph construction is an important step in graph-based data analysis. It involves determining the optimal way to represent data as nodes (the points of a graph) and edges (the connections between nodes), ensuring the graph is relevant and effective for the specific research question being addressed. Some graph constructions are based on domain knowledge and objective fact, such as citation graphs (100), gene regulatory networks (101), and protein interaction networks (102). However, this domain knowledge is not always readily available. For example, fewer than 5% of human protein interactions have been structurally characterized (103). As a result, data-driven graph construction often becomes essential and can provide novel insights to researchers. A data-driven graph construction algorithm refers to methods that build graphs based on the distribution of raw data and employ techniques, such as statistical methods, without heavily relying on predefined domain knowledge.

Data-driven graph construction methods typically unfold in two essential steps that critically shape the structure and effectiveness of the resulting graph. The first step involves calculating either dissimilarity measures (such as Manhattan or Euclidean distance) or similarity scores (e.g., Spearman, Pearson, or Cosine correlation) between nodes. This foundational step quantifies relationships based on node characteristics, laying the groundwork for meaningful connections. The second step focuses on constructing edges between nodes based on these computed scores. There are two main approaches: one applies a global threshold across the entire graph to determine whether an edge should be retained, while the other leverages the local topological structure to make this decision (104). In the life sciences, previous research has explored methods for selecting an appropriate global threshold for gene expression data (48). However, comparatively little attention has been given to graph construction methods in genomic data that consider the relative weights of edges with respect to their neighboring edges. This paper proposes a novel graph

construction method that relies on the local topological structure of vertices and explores its potential application in the analysis of gene expression data.

In Section 4.2, we present the proposed graph construction algorithm and state and prove several theorems describing the properties of the resulting graphs. In Section 4.3, we apply our algorithm to several gene expression datasets and assess the quality of the constructed graphs through vertex clustering performance. Section 4.4 highlights the advantages of our approach and outlines potential directions for future research. For clarity, we provide below the definitions of the symbols used throughout this chapter:

Symbols:

- cor_{ij} - Correlation between vertices v_i and v_j .
- e_{ij} - The edge between vertices i and j . In an unweighted graph, a ‘1’ indicates the presence of an edge, while a ‘0’ indicates the absence of an edge.
- d_{ij} - The dissimilar score between vertices i and j
- $N(v_i)$ - The set of vertices adjacent to vertex i , referred to as its neighbors.
- v_i - The vertex i

4.2 Power Gabriel Graph

Proximity graphs are essential data structures in computational geometry, capturing proximity relationships between points, usually defined in terms of a distance metric (105). Gabriel graphs are a type of proximity graph where edges are established based on criteria delineated by the equation 4.1.

$$e_{ij} = \begin{cases} 1 & \text{if } \forall p \neq i, j, \quad (d_{ij})^2 \leq (d_{ip})^2 + (d_{jp})^2, \\ 0 & \text{otherwise.} \end{cases} \quad (4.1)$$

Originally introduced in 1969, the Gabriel Graph was developed to construct geographic-connectivity graphs, in which vertices represent geographic locations (106). Its structure captures spatial relationships between locations, allowing researchers to analyze how geographic factors influence patterns of biological variation. In this century, it has found applications in diverse areas, such as analyzing high-dimensional data (107), enhancing sensor network localization (108), examining network resilience—particularly in modeling physical-level networks with grid-like properties (109)—and the semi-supervised estimation of cell-state transitions in single cell (110). The Gabriel Graph has a variant called the k -Gabriel Graph (111), where vertices i and j are connected by an edge if fewer than k vertices in the graph violate Equation 4.1. This variant has been applied to improve solutions for the Euclidean bottleneck biconnected edge subgraph problem (111), in studies of Hamiltonian properties (112), and in other related areas.

Based on our literature review, the Gabriel Graph has not been widely used in gene expression data analysis. However, we were inspired by its approach to graph construction—specifically, its use of local comparisons between adjacent edges rather than applying a global threshold across all edges. In this paper, we propose an enhanced variant of the traditional Gabriel Graph, which we call the Power Gabriel Graph (PGG). The PGG modifies the edge selection criteria by introducing a power parameter, n , which enables adjustable sensitivity to the proximity between points. This gives researchers greater flexibility to control how strictly edges are filtered, as well as to adjust the overall density of the resulting graph. In our approach, gene dissimilarity is represented as one minus the absolute value of the Pearson correlation, following common practice in gene expression data analysis, where Pearson correlation is typically used to measure gene similarity (48). The input to the PGG algorithm is a simple, undirected, edge weighted graph with vertices such as v_i , v_j , and v_p . The output is an unweighted graph in which edges are retained according to the following criteria:

$$e_{ij} = \begin{cases} 1 & \text{if } \forall p \in N(v_i) \cap N(v_j), (1 - |\text{cor}_{ij}|)^n \leq (1 - |\text{cor}_{ip}|)^n + (1 - |\text{cor}_{jp}|)^n, \\ 0 & \text{otherwise.} \end{cases} \quad (4.2)$$

4.2.1 Key Properties of the Power Gabriel Graph

To simplify the proof process, we define the vertex dissimilarity value $d_{ij} = 1 - |\text{cor}_{ij}|$ for any pair of vertices i and j . When using Pearson correlation coefficients, the value of cor_{ij} lies between -1 and 1 , so $1 - |\text{cor}_{ij}|$ is a non-negative number in the interval $[0, 1]$.

Theorem 4.1. *Let G_n be a graph generated by the Power Gabriel Graph algorithm with power n , and let E_n denote the set of edges in G_n . Then, for any $n_1 > n_2$, it holds that $E_{n_1} \subseteq E_{n_2}$. In other words, increasing the power results in the removal of edges only; no new edges are added.*

Proof. It is known that d_{ij} , d_{ip} , and d_{jp} are all non-negative numbers and $n > 1$ is a positive integer. As a first step, we aim to prove that if

$$d_{ij}^n \leq d_{ip}^n + d_{jp}^n,$$

then

$$d_{ij}^{n-1} \leq d_{ip}^{n-1} + d_{jp}^{n-1}.$$

Without loss of generality, we can always assume $d_{ip} \leq d_{jp}$. It is discussed in two cases.

Case 1. $d_{ij} \leq d_{jp}$.

In this case, $d_{ij}^{n-1} \leq d_{jp}^{n-1}$. obviously,

$$d_{ij}^{n-1} \leq d_{ip}^{n-1} + d_{jp}^{n-1}.$$

Case 2. $d_{ij} > d_{jp}$.

Then $d_{ij} > d_{ip}$. If

$$d_{ij}^n \leq d_{ip}^n + d_{jp}^n,$$

then

$$d_{ij}^{n-1} \leq \frac{d_{ip}^n + d_{jp}^n}{d_{ij}}.$$

And because of

$$d_{ip}^n + d_{jp}^n = d_{ip}d_{ip}^{n-1} + d_{jp}d_{jp}^{n-1} < d_{ij}d_{ip}^{n-1} + d_{ij}d_{jp}^{n-1} = d_{ij}(d_{ip}^{n-1} + d_{jp}^{n-1}),$$

$$d_{ij}^{n-1} < \frac{d_{ij}(d_{ip}^{n-1} + d_{jp}^{n-1})}{d_{ij}} = d_{ip}^{n-1} + d_{jp}^{n-1}.$$

We have shown that if $d_{ij}^{n_1} \leq d_{ip}^{n_1} + d_{jp}^{n_1}$, then the inequality also holds for $n_1 - 1$. Consequently, we have $E_{n_1} \subseteq E_{n_1-1}$. By applying this argument iteratively, we conclude that

$$E_{n_1} \subseteq E_{n_1-1} \subseteq \cdots \subseteq E_{n_2+1} \subseteq E_{n_2}$$

whenever $n_1 > n_2$, thereby establishing Theorem 4.1.

Theorem 4.2. *If the input graph contains no isolated vertices, then the graph constructed by the Power Gabriel Graph algorithm also contains no isolated vertices for all power values.*

Proof. For any vertex i , let $d_{ij} = \min\{d_{ip} \mid v_p \in N(v_i)\}$, where $N(v_i)$ denotes the set of neighbors of v_i prior to applying the Power Gabriel Graph algorithm. Then, for all $v_p \in N(v_i)$, we have

$$(d_{ij})^n \leq (d_{ip})^n \leq (d_{ip})^n + (d_{jp})^n,$$

since d_{ij} , d_{ip} , and d_{jp} are all non-negative and n is a positive value. In other words, each vertex v_i will have an edge to the vertex v_j with which it has the strongest absolute correlation.

Corollary 4.2.1. *If the input graph contains no isolated vertices, then the lower bound of the density of the graph constructed by the Power Gabriel Graph algorithm is $\frac{1}{|V|-1}$, where $|V|$ is the number of vertices.*

Proof. Theorem 4.2 proves that a graph G constructed by the Power Gabriel Graph algorithm has no isolated vertices. Therefore, the minimum possible number of edges in G is $\lceil \frac{|V|}{2} \rceil$. The corresponding minimum possible density is

$$\frac{\lceil |V|/2 \rceil}{|V|(|V|-1)/2}.$$

Thus, $\frac{1}{|V|-1}$ serves as a lower bound for the density of graphs constructed by the Power Gabriel Graph algorithm.

It is worth noting that the proofs of Theorems 4.1, 4.2, and Corollary 4.2.1 only require that $d_{ij} > 0$ and do not depend on d_{ij} being confined to the interval $[0, 1]$. Therefore, if the Power Gabriel Graph is adapted for use in other domains where vertex similarity is not computed using Pearson correlation, these theoretical properties still hold—as long as the vertex dissimilarity values remain positive.

4.2.2 Time Complexity of Constructing a Power Gabriel Graph

From Equation (4.2), it is evident that for each pair of vertices, their correlation scores must be compared with all other vertices that have correlation values with both vertices. This evaluation involves comparing correlations with up to $O(|V|)$ other vertices, resulting in a per-pair complexity of $O(|V|)$. Since there are $O(|V|^2)$ vertex pairs in total, the overall time complexity of constructing the Power Gabriel Graph is $O(|V|^3)$, where $|V|$ is the number of vertices. We have developed a C++ package to perform this computation, along with a Python package that supports parallel execution. Since the Power Gabriel Graph algorithm executes at an acceptable speed, speed optimization will not be a focus in the subsequent experiments.

4.3 Implementation and Testing

To validate the performance of the Power Gabriel Graph algorithm, we tested it on multiple gene expression datasets. We compared it with the traditional single global thresholding method and also explored a combined approach that integrates both methods. Furthermore, we investigated how vertex clustering performance is affected by varying the power parameter n .

4.3.1 Datasets and Graph Construction

The datasets used in this study were all gene expression datasets obtained from the Gene Expression Omnibus (GEO) database (113). To assess the robustness of our algorithm, we conducted separate evaluations on the *Homo sapiens* and *Mus musculus* datasets. Each dataset contained multiple samples with corresponding gene expression probe values. A summary of the dataset sizes and key characteristics is provided in the tables in Appendix B. Gene expression values were derived from probe-level data using the GEOquery R package (114). For genes represented by multiple probes, the mean expression value across probes was used to generate a single gene-level expression measurement.

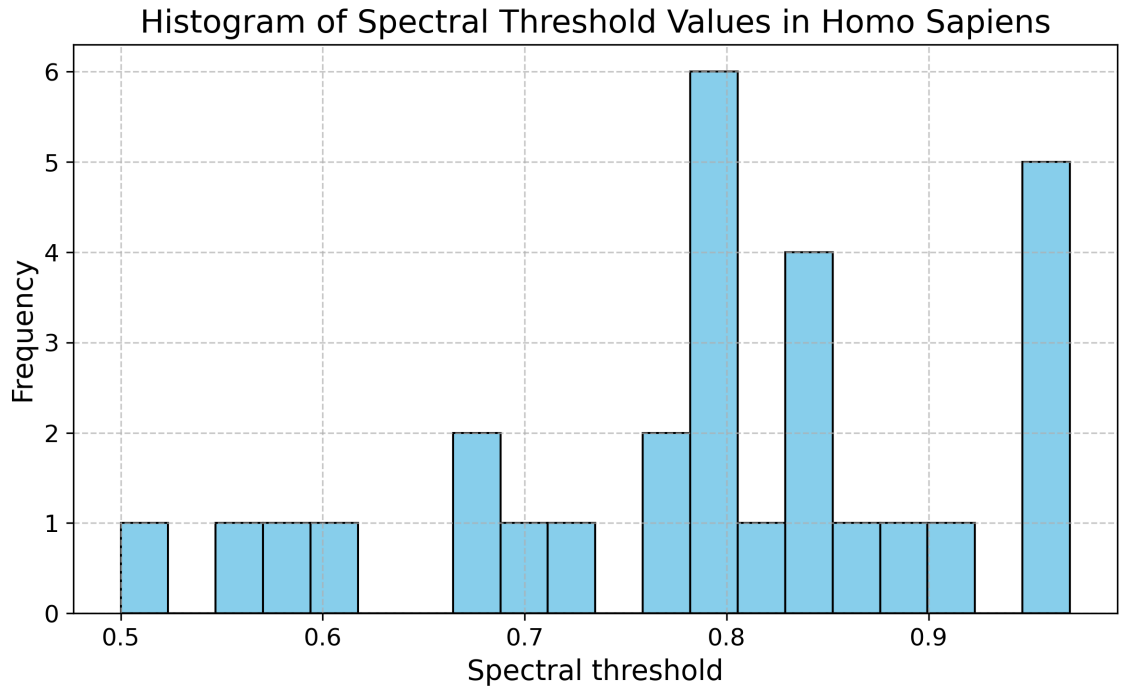
For each *Homo sapiens* dataset, which included at least two subtypes (e.g., patients vs. controls), we constructed separate graphs for each subtype by using genes as vertices and defining edge weights as the absolute values of Pearson correlation coefficients computed from gene expression values across the samples belonging to that subtype. For the *Mus musculus* datasets, graphs were similarly constructed using genes as vertices and Pearson correlation-based edge weights, calculated across all samples in the dataset. In total, we generated 29 initial graphs for *Homo sapiens* and 18 for *Mus musculus*. Prior to applying data-driven graph construction algorithms, we conducted a preprocessing step to enhance computational efficiency, given that human and mouse datasets typically include over 20,000 genes and yield large, resource-intensive graphs. For *Homo sapiens*, edges with weights below 0.5 were removed, and

any resulting isolated vertices were excluded. For *Mus musculus*, a stricter threshold of 0.6 was used for edge removal, followed by the same pruning of isolated vertices.

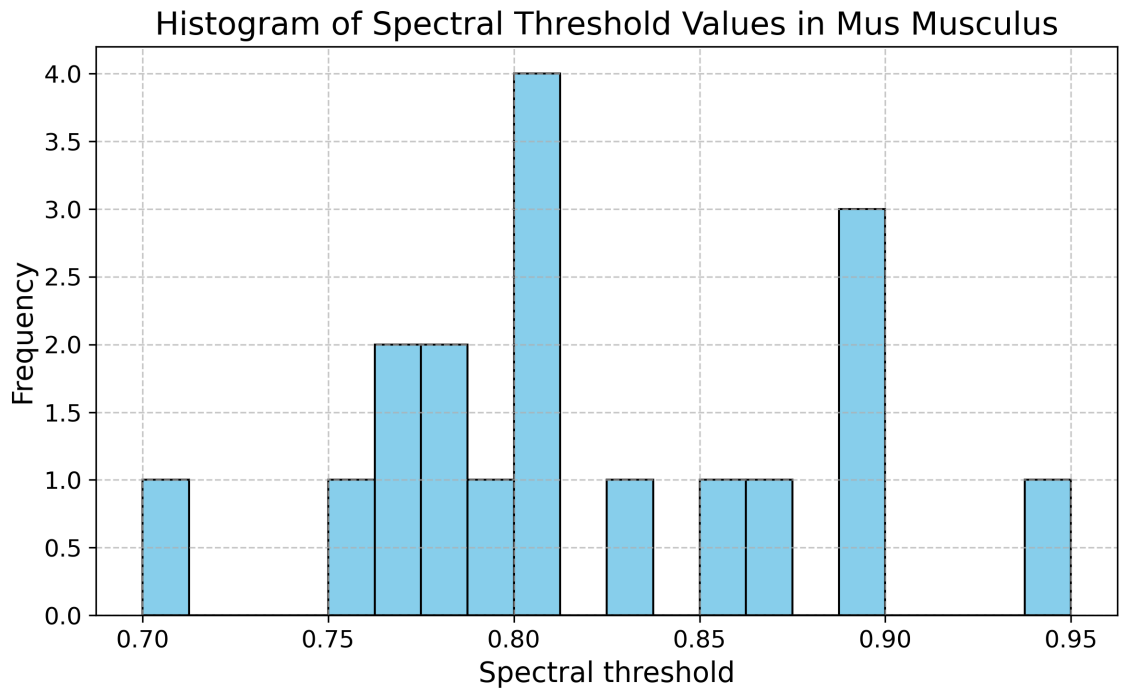
After preprocessing, we employed three data-driven methods to construct the graphs: (1) the spectral graph theory algorithm, a widely used global thresholding technique in genomic data analysis (115); (2) the Power Gabriel Graph algorithm; and (3) the Power Gabriel Graph algorithm applied to the spectral-thresholded graphs (spectral + PGG). In Method (1) and (3), to determine the optimal global spectral threshold while minimizing computation time, we first performed a coarse search by testing thresholds ranging from 0.5 to 1.0 in increments of 0.05 to get a initial threshold. We then performed a finer search to refine this estimate within the range $\max(0.5, \text{initial threshold} - 0.05)$ to $\min(1, \text{initial threshold} + 0.05)$, using a step size of 0.01. Figure 4.1 shows the distribution of thresholds selected by the spectral graph thresholding method. In Method (2), we used the graphs generated after the preprocessing steps as input graphs. In Method (3), we used the graphs from Method (1) as input graphs. For both Method (2) and Method (3), we then applied the Power Gabriel Graph criterion, defined in Equation 4.2, with the power parameter n ranging from 1 to 10 to construct a series of ten additional graphs for each input graph.

4.3.2 Graph Vertex Clustering Analysis

We compared the graphs constructed by the three methods described in the previous section by evaluating how variations in graph construction criteria—and the resulting structural differences—affect vertex clustering performance. Following the gene vertex clustering and evaluation workflow used in the global thresholds study by (48), we investigated whether our Power Gabriel Graph approach offers advantages for constructing gene graphs. For each graph constructed using the three methods described in Section 4.3.1, we applied the paraclique algorithm (116) to cluster the gene vertices. The paraclique algorithm clustered vertices by first identifying a maximum clique, then expanding it to include additional vertices that were highly



(a) Spectral thresholds of *Homo sapiens* graphs

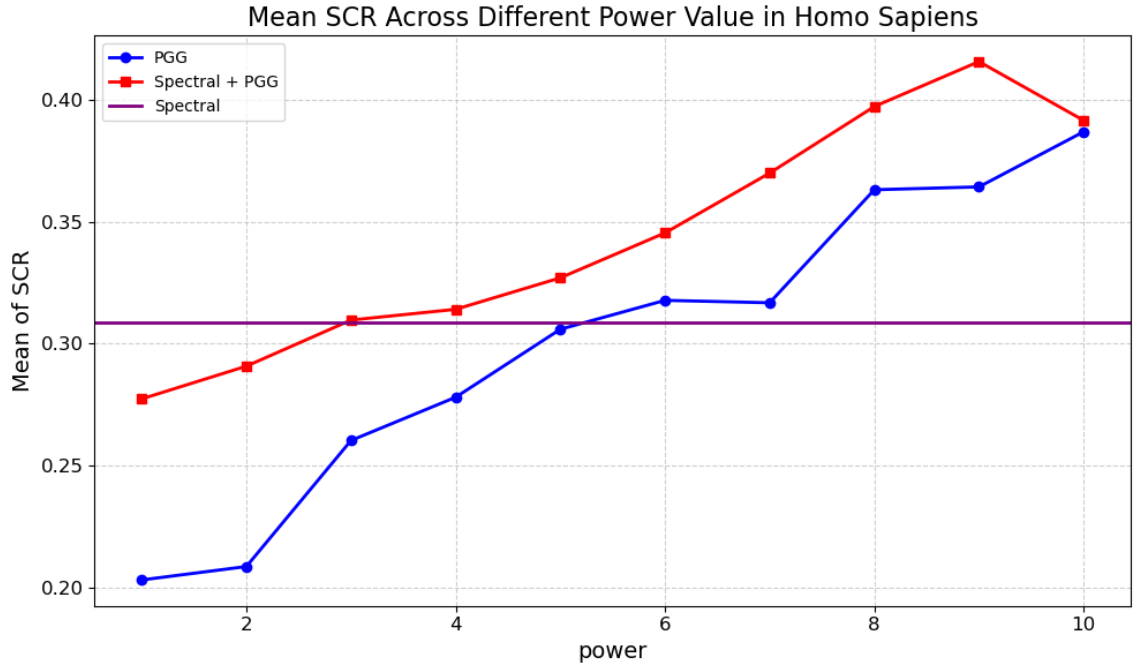


(b) Spectral thresholds of *Mus musculus* graphs

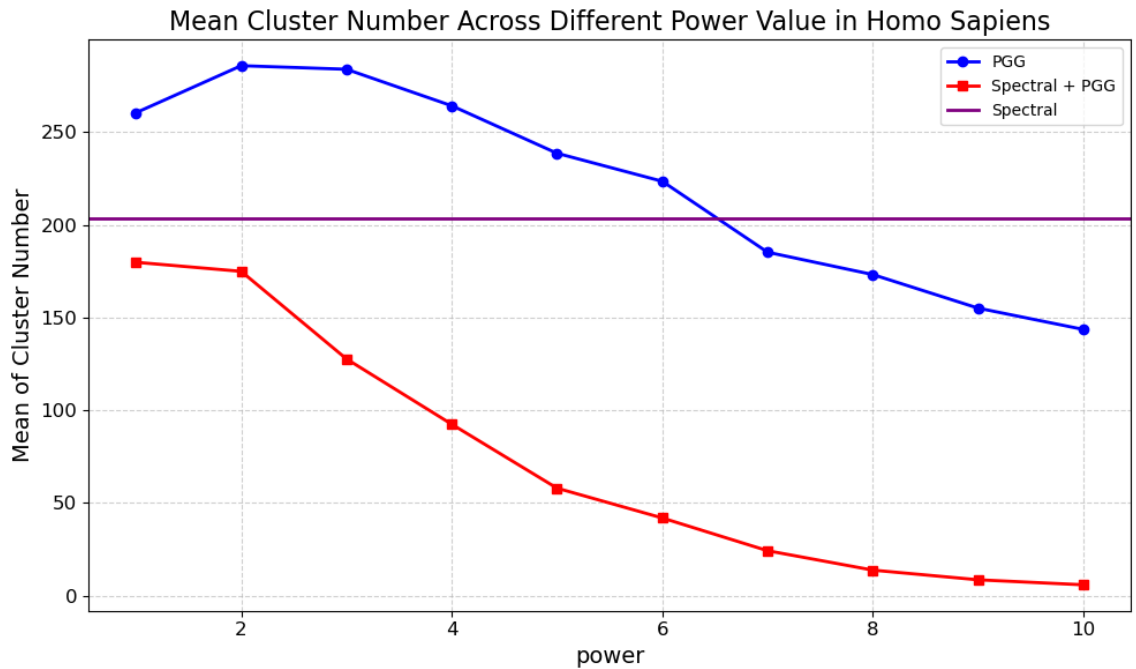
Figure 4.1: Histogram of spectral graph theory thresholds values of test graphs.

connected to the clique. This expansion allowed newly added vertices to be connected to all but a limited number of existing paraclique members, with the maximum number of allowable missing edges defined by a parameter known as the glom term. We used the following parameters during the clustering process: glom term of 1, a minimum paraclique size of 4, and a maximum of 300 generated paracliques (117). We then performed Gene Ontology (GO) biological process enrichment analysis on each cluster using the GSEAPy package (118). A cluster was considered significant if its GO enrichment results included at least one enriched GO term with an adjusted p -value less than 0.01. Finally, we evaluated the quality of each graph using the Significant Clusters Ratio (SCR) (48), defined as the number of significant clusters divided by the total number of clusters produced by the paraclique algorithm. It is worth noting that the paraclique step can be time-consuming, so we set a timeout of 24 hours. If a graph could not be processed within this time window, or if the paraclique algorithm failed to identify any clusters, the graph was marked as having no valid SCR.

We analyzed the *Homo sapiens* and *Mus musculus* datasets separately. For each species, we calculated the average SCR of the spectral-thresholded graphs to serve as a baseline. For graphs constructed using the PGG algorithm and the spectral + PGG approach, we computed the average SCR at each power value and compared these values to the corresponding species-specific baseline. Figure 4.2a shows how the average SCR changes across power values of *Homo sapiens* data. When using the PGG algorithm alone, the average SCR increased with the power value, surpassing the spectral-thresholded baseline when the power exceeded 5. Similarly, the spectral + PGG method also showed increasing SCR with higher power values, though a slight decline was observed when the power increased from 9 to 10. This combined method consistently achieved higher average SCR than the PGG-only approach at the same power level and exceeded the spectral baseline when the power reached 3. Figure 4.2b illustrates how the average number of clusters changed with increasing power of *Homo sapiens* data. As expected, the number of clusters decreased once



(a) Average SCR across power values in *Homo sapiens*



(b) Average cluster number across power values in *Homo sapiens*

Figure 4.2: Mean significant clusters ratio (SCR) and number of clusters across different power values in *Homo sapiens* test data. Each point represents the average SCR or cluster number across all graphs for a given power value within the species. The curve of mean SCR values shows a generally increasing trend, with some fluctuations. When the power value exceeded 2, the average number of clusters progressively decreased.

Table 4.1: Graph counts where one method outperforms another in SCR across power settings for *Homo sapiens* data

Power	PGG > Spectral	Spectral+PGG > Spectral	PGG > Spectral+PGG
1	2/16	12/29	5/16
2	5/29	13/29	3/29
3	11/29	15/29	8/29
4	16/29	15/29	12/29
5	17/28	14/28	13/27
6	17/28	14/26	13/25
7	15/28	14/24	11/23
8	21/29	11/22	10/22
9	20/29	12/20	8/20
10	22/29	12/20	8/20

the power value reached 2 or higher, due to the reduced graph density. Additionally, graphs generated using the PGG algorithm alone exhibited more clusters compared to those constructed with the spectral + PGG approach.

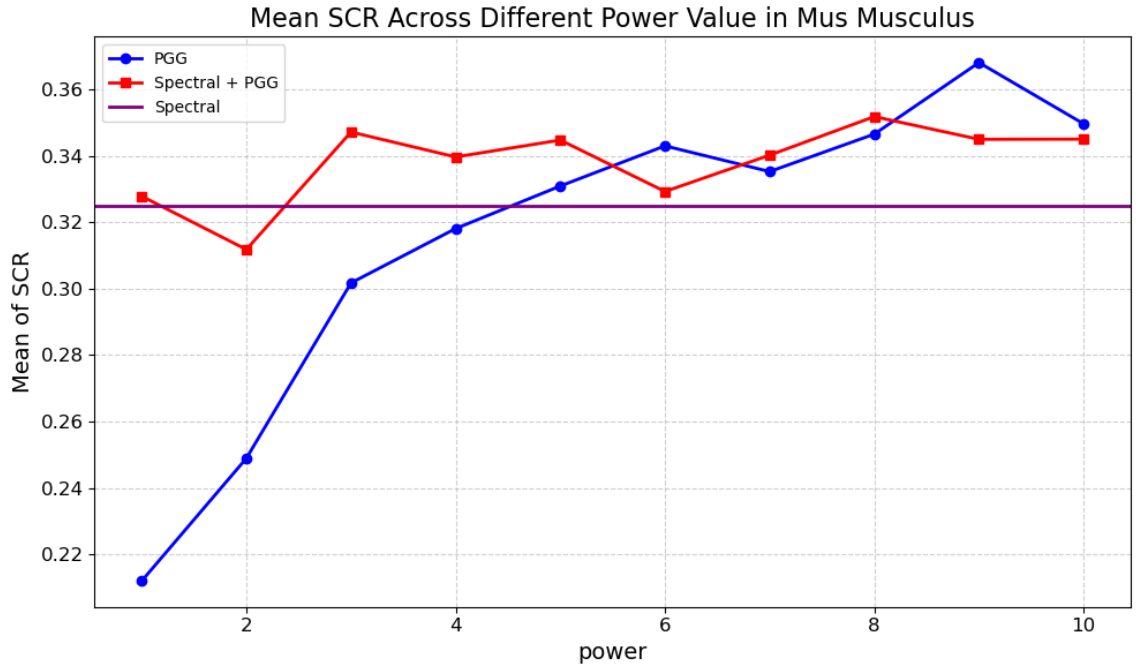
In addition to comparing average SCR values, we conducted pairwise comparisons between the three methods by evaluating the SCR of graphs constructed from the same initial graphs. For example, when comparing the PGG method with the spectral-thresholded method, we first identified, for each power value, the number of (PGG, spectral) graph pairs where both graphs were constructed from the same initial graph and had valid SCR values. We then counted how many of these pairs showed higher SCR in the PGG graph compared to its spectral-thresholded counterpart. We performed similar comparisons between spectral + PGG and spectral-only methods, as well as between PGG and spectral + PGG. The results are summarized in Table 4.1. As shown, at a power value of 4, the PGG graphs outperform their spectral-thresholded counterparts in 16 out of 29 cases. This trend continues at higher power values, where a majority of PGG graphs consistently achieve higher SCR scores. The Spectral+PGG method begins to outperform the Spectral-only approach when the

power exceeds 3, with more than half of the graphs showing improved SCR. These results highlight the effectiveness of both PGG and Spectral+PGG in constructing high-quality *Homo sapiens* gene graphs, providing researchers with flexible options depending on their analytical objectives.

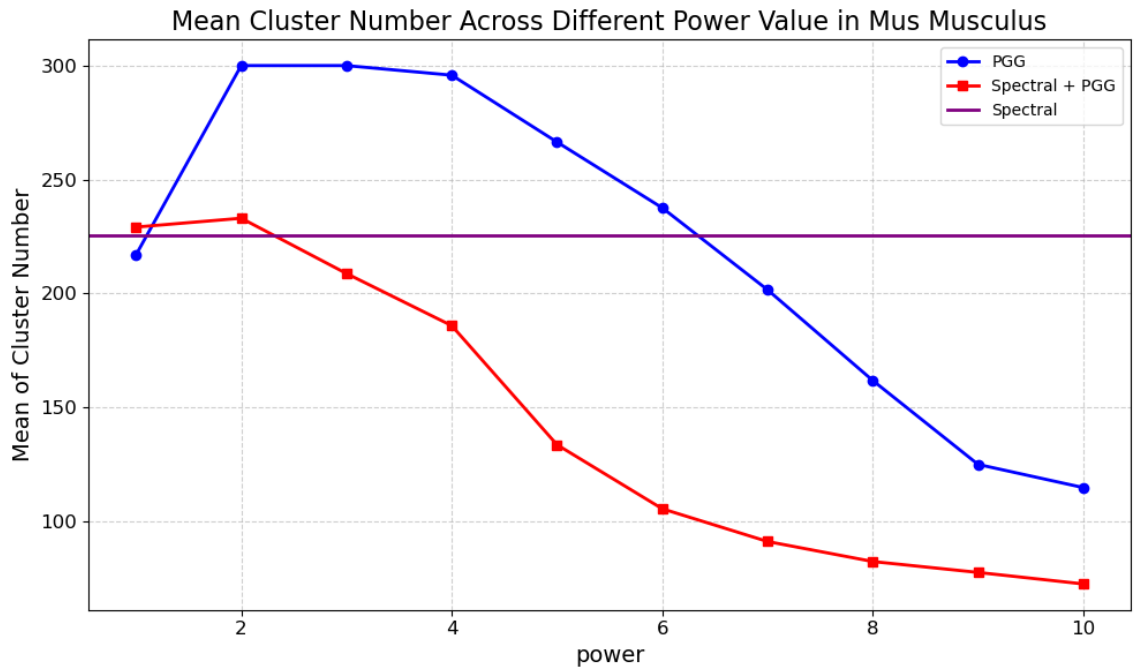
Figure 4.3 shows how the average SCR and the average number of clusters of *Mus musculus* data change across different power values, compared with the spectral-thresholded graphs. The trend of the average SCR curve for the PGG-only method in *Mus musculus* is similar to that observed in *Homo sapiens*. The average SCR increases with the power value and surpasses the spectral-thresholded baseline when the power reaches 5. When applying the spectral + PGG method to *Mus musculus* data, the average SCR exceeds that of the spectral-thresholded graphs at all tested power values, except at power 2. We conducted a pairwise comparison between the three graph construction methods, following the same procedure used for the *Homo sapiens* data. The results are summarized in Table 4.2. When the power value reaches 6, more than half of the PGG graphs (9 out of 17) exhibit higher SCR than the spectral-thresholded graphs. For all power values above 6, this trend persists, with the majority of PGG graphs consistently outperforming their spectral-thresholded counterparts. Likewise, when comparing the Spectral+PGG method to the spectral-only approach, we observe that—except at power level 2—more than half of the Spectral+PGG graphs achieve higher SCR values across all remaining power levels.

4.4 Discussion

In this study, we introduced a novel graph construction algorithm—Power Gabriel Graph—an extension of the standard Gabriel Graph that incorporates a tunable power parameter to control edge formation. We demonstrated its advantages in a bioinformatics context. This method addresses a key limitation in data-driven network construction within the life sciences: whereas most existing approaches apply a single global threshold across all edges, PGG leverages local topological structure



(a) Average SCR across power values in *Mus musculus*



(b) Average cluster number across power values in *Mus musculus*

Figure 4.3: Mean significant clusters ratio (SCR) and number of clusters across different power values in *Mus musculus* test data. Each point represents the average SCR or cluster number across all graphs for a given power value within the species.

Table 4.2: Graph counts where one method outperforms another in SCR across power settings for *Mus musculus* data

Power	PGG > Spectral	Spectral+PGG > Spectral	PGG > Spectral+PGG
1	0/8	12/18	0/7
2	3/18	7/18	1/17
3	6/18	11/18	2/17
4	6/18	13/18	5/17
5	7/17	12/18	7/16
6	9/17	11/18	8/16
7	9/16	12/18	6/15
8	9/16	12/18	6/15
9	11/15	11/18	8/14
10	7/12	11/18	5/11

to enable more adaptive and context-sensitive edge filtering. Furthermore, as the power parameter increases, PGG produces progressively sparser graphs while avoiding the introduction of isolated vertices, provided the input graph contains none. This approach offers a controllable mechanism for adjusting graph density while largely reinforcing local nearest-neighbor relationships.

Second, we applied PGG to gene expression data—a domain in which Gabriel Graphs have historically seen limited application—and demonstrated its potential to enhance the construction of gene co-expression networks. We evaluated the algorithm using datasets from *Homo sapiens* and *Mus Musculus*. Both PGG and the combined spectral + PGG approach provided added value compared to using the spectral method alone. Notably, in *Homo sapiens* and *Mus Musculus*, the mean SCR curve exhibited an increasing trend as the power parameter increased. These results suggest that PGG can uncover biologically meaningful patterns that may be missed by conventional single threshold-based methods.

Except for *Homo sapiens* and *Mus musculus*, we also evaluated the PGG algorithm on *Drosophila melanogaster* and *Saccharomyces cerevisiae*. However, the

performance curves observed in these species did not replicate the patterns seen in *Homo sapiens* and *Mus musculus*. In particular, the best performance for both *Drosophila melanogaster* and *Saccharomyces cerevisiae* was achieved at low power values, such as one or two. One possible explanation is that, in *Homo sapiens* and *Mus musculus* experiments, samples are generally obtained from specific tissues—such as brain or skin—whereas sample collection protocols for *Drosophila melanogaster* and *Saccharomyces cerevisiae* differ substantially. These findings suggest that the PGG algorithm attains optimal performance at different power levels depending on the dataset.

Chapter 5

Conclusions and New Research Directions

This dissertation investigates and advances graph-theoretical algorithms with the dual objectives of contributing to fundamental graph theory and addressing complex problems in the life sciences. It presents three core contributions:

Chapter 2 introduces a novel multipartite link prediction algorithm that uncovers hidden relationships within disease–drug–protein networks. This method simultaneously identifies candidate disease–drug, disease–protein, and drug–protein links, offering promising leads for drug repurposing and deeper insights into polypharmacology. The results highlight the power of heterogeneous networks in biomedical applications.

We point out that this work may open a few new research directions. For example, investigators might want to anchor augmentation on a poorly understood disease or other targeted vertex of special interest. They could also try to incorporate this methodology with other drug repositioning techniques, and in fact such an integrative approach seems to have become increasingly popular (119; 120; 121; 122). Finally, researchers might wish to add highly relevant partite sets denoting categories such as side effects, or they may seek to test the effectiveness of a k -clique augmentation

strategy in the context of closely related problem domains such as studies of drug-drug and protein-protein interaction networks.

Chapter 3 presents a faster k -partite clique enumeration algorithm, MPCE, that efficiently identifies all maximal k -partite cliques. Experiments on both synthetic and real-world datasets demonstrate its significant performance gains, effectively extending the tractable limits of this $\mathcal{NP}$ -hard problem. This improvement also enables other algorithms—such as the link prediction method in Chapter 2—to scale to larger and more complex biological networks. To scale the algorithm for even larger and more heterogeneous graphs, future research could pursue several promising directions, including parallelizing its implementation.

Parallelizing the enumeration of maximal or other cliques presents numerous challenges, particularly due to the inherent complexity of distributing the workload across multiple processors (40). One significant issue is the tendency for parallel processes to result in highly unbalanced execution times, as the search tree generated by clique enumeration algorithms is often irregular and unpredictable. Additionally, naive dynamic load distribution strategies can be prohibitively expensive because they require frequent and substantial data movement between processors, further complicating the parallelization process.

Despite these challenges, some researchers have explored parallel approaches to speed up the enumeration process in general graphs. For instance, efforts have been made to use shared memory architectures (123; 40), minimizing the transfer of large datasets between processors (40), and applying graph partitioning techniques (124). Furthermore, efforts have been made to find maximum cliques in general graphs through parallelization (43; 44) or hybrid approaches to address the challenges of parallelization (45; 46). There have also been a few studies on parallelizing the process of finding k -cliques (125) and quasi-cliques (126). However, research focused on the parallel enumeration of maximal k -partite cliques remains limited and represents a promising direction for future work.

Advances in GPUs have enabled researchers to develop GPU-based Bron-Kerbosch algorithms that support parallel execution (127). Since our algorithm is also based on the Bron-Kerbosch framework, enabling GPU parallelism for MPCE could be a significant step forward. We believe this could be a feasible direction for future development, potentially leading to substantial improvements in performance and scalability.

Chapter 4 proposes a new graph construction strategy, PGG, for generating reliable and interpretable networks from gene expression data. By incorporating adjustable, local edge-weight-based filtering, this method overcomes limitations of traditional approaches that rely on a single global threshold and often overlook local structures. The results of the vertex clustering tasks support the conclusion that the new graph construction strategy enables more accurate and meaningful biological network analyses.

In our experiments, PGG achieved the best SCR at different power values across species. Moreover, for each species, multiple power values yielded average SCR scores that surpassed those of the spectral-thresholded graphs. It is important to note that SCR is a metric computed after graph construction, rather than one used before applying the graph construction algorithms. In light of these considerations, developing systematic methods for selecting optimal power values remains a valuable avenue for future research.

Last but not least, although all the graphs discussed in this dissertation are finite, simple, static, and undirected, more complex graph structures—particularly dynamic and directed graphs—also arise in real-world applications. Dynamic maximal cliques have been employed in online social networks to provide valuable social intelligence (128), and have also been applied in dynamic graphs for node embedding tasks (129). Moreover, dynamic k -partite graphs have been utilized in areas such as selecting access networks with improved quality of service in heterogeneous wireless environments (130). Analyses involving regulatory networks and signaling pathways often rely on directed graphs. Adapting the algorithms developed in this dissertation to support

such graph types could significantly enhance their applicability and represents a promising direction for future research.

Bibliography

- [1] L. Euler, “Leonhard euler and the königsberg bridges,” *Scientific American*, vol. 189, no. 1, pp. 66–72, 1953. [1](#)
- [2] M. Rigo, *Advanced graph theory and combinatorics*. John Wiley & Sons, 2016. [2](#)
- [3] K. R. Saoub, *Graph Theory: an introduction to proofs, algorithms, and applications*. Chapman and Hall/CRC, 2021. [2](#)
- [4] E. Estrada, “Graph and network theory,” *Glasgow: University of Strathclyde*, 2013. [2](#)
- [5] A. G. Marques, S. Segarra, and G. Mateos, “Signal processing on directed graphs: The role of edge directionality when processing and learning from network data,” *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 99–116, 2020. [2](#)
- [6] Y.-Y. Feng, H. Yu, Y.-H. Feng, and J.-Y. Shi, “Directed graph attention networks for predicting asymmetric drug–drug interactions,” *Briefings in Bioinformatics*, vol. 23, no. 3, p. bbac151, 2022. [2](#)
- [7] C.-X. Lin, T.-W. Huang, and M. D. Wong, “An efficient work-stealing scheduler for task dependency graph,” in *2020 IEEE 26th international conference on parallel and distributed systems (ICPADS)*, pp. 64–71, IEEE, 2020. [2](#)

- [8] H. F. Halaoui, “Intelligent traffic system: road networks with time-weighted graphs,” *International Journal for Infonomics*, vol. 3, no. 4, pp. 350–359, 2010. [2](#)
- [9] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete, “Weighted graph comparison techniques for brain connectivity analysis,” in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 483–492, 2013. [2](#)
- [10] M. Kargar, L. Golab, D. Srivastava, J. Szlichta, and M. Zihayat, “Effective keyword search over weighted graphs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 2, pp. 601–616, 2020. [2](#)
- [11] P. Jaccard, “Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines,” *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901. [2](#)
- [12] L. Adar and L. Adamic, “Friends and neighbors on the web,” *social networks*, vol. 25, no. 3, pp. 211–230, 2003. [2](#)
- [13] A.-L. Barabási, H. Jeong, Z. Nédá, E. Ravasz, A. Schubert, and T. Vicsek, “Evolution of the social network of scientific collaborations,” *Physica A: Statistical mechanics and its applications*, vol. 311, no. 3-4, pp. 590–614, 2002. [2](#)
- [14] L. Getoor, N. Friedman, D. Koller, and B. Taskar, “Learning probabilistic models of link structure,” *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 679–707, 2002. [2](#)
- [15] S. Gu, L. Chen, B. Li, W. Liu, and B. Chen, “Link prediction on signed social networks based on latent space mapping,” *Applied Intelligence*, vol. 49, pp. 703–722, 2019. [3](#)

- [16] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, “Link prediction techniques, applications, and performance: A survey,” *Physica A: Statistical Mechanics and its Applications*, vol. 553, p. 124289, 2020. [3](#)
- [17] Z. Wu and Y. Chen, “Link prediction using matrix factorization with bagging,” in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pp. 1–6, IEEE, 2016. [3](#)
- [18] B. Chen, F. Li, S. Chen, R. Hu, and L. Chen, “Link prediction based on non-negative matrix factorization,” *PloS one*, vol. 12, no. 8, p. e0182968, 2017. [3](#)
- [19] V. P. Chaubey, A. Sharma, T. Sharma, S. Sharma, and A. Kumar, “Link prediction for social network analysis using random forest and xg-boost algorithm,” in *2023 Second International Conference on Informatics (ICI)*, pp. 1–6, IEEE, 2023. [3](#)
- [20] S. Kumar, A. Mallik, and B. Panda, “Link prediction in complex networks using node centrality and light gradient boosting machine,” *World Wide Web*, vol. 25, no. 6, pp. 2487–2513, 2022. [3](#)
- [21] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” *Advances in neural information processing systems*, vol. 31, 2018. [3](#)
- [22] L. Cai, J. Li, J. Wang, and S. Ji, “Line graph neural networks for link prediction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5103–5113, 2021. [3](#)
- [23] Q. Liu, Y.-P. P. Chen, and J. Li, “k-partite cliques of protein interactions: A novel subgraph topology for functional coherence analysis on ppi networks,” *Journal of theoretical biology*, vol. 340, pp. 146–154, 2014. [4](#), [32](#)
- [24] C. Chen, S. K. Grady, S. R. Ellingson, and M. A. Langston, “Gene-disease-drug link prediction using tripartite graphs,” in *Proceedings of the 12th ACM*

- Conference on Bioinformatics, Computational Biology, and Health Informatics*, pp. 1–1, 2021. [4](#)
- [25] S. Chen, F. D. Bowman, and Y. Xing, “Detecting and testing altered brain connectivity networks with k-partite network topology,” *Computational statistics & data analysis*, vol. 141, pp. 109–122, 2020. [4](#), [6](#), [32](#)
- [26] W. Chen, W. Hendrix, and N. F. Samatova, “The application of the weighted k-partite graph problem to the multiple alignment for metabolic pathways,” *Journal of Computational Biology*, vol. 24, no. 12, pp. 1195–1211, 2017. [4](#)
- [27] E. Seo, D. Kang, and H.-J. Choi, “Metadata-based collaborative filtering using k-partite graph for movie recommendation,” in *The Fourth International Conference on Emerging Databases-Technologies, Applications, and Theory (EDB)*, pp. 54–61, EDB, 2012. [5](#)
- [28] A. Zhou, Y. Wang, and L. Chen, “Finding large diverse communities on networks: The edge maximum k*-partite clique,” *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2576–2589, 2020. [5](#), [6](#), [32](#)
- [29] P. Kefalas, P. Symeonidis, and Y. Manolopoulos, “A graph-based taxonomy of recommendation algorithms and systems in lbsns,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 604–622, 2015. [5](#)
- [30] Y. Gao, M. Wang, R. Ji, Z. Zha, and J. Shen, “k-partite graph reinforcement and its application in multimedia information retrieval,” *Information Sciences*, vol. 194, pp. 224–239, 2012. [5](#)
- [31] B. Long, X. Wu, Z. Zhang, and P. S. Yu, “Unsupervised learning on k-partite graphs,” in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 317–326, 2006. [5](#)
- [32] R. Amin, N. Shah, and W. Mehmood, “Enforcing optimal acl policies using k-partite graph in hybrid sdn,” *Electronics*, vol. 8, no. 6, p. 604, 2019. [5](#)

- [33] T. Grünert, S. Irnich, H.-J. Zimmermann, M. Schneider, and B. Wulfhorst, “Finding all k-cliques in k-partite graphs, an application in textile engineering,” *Computers & operations research*, vol. 29, no. 1, pp. 13–31, 2002. [5](#), [6](#), [32](#)
- [34] Z. Ali, G. Qi, P. Kefalas, W. A. Abro, and B. Ali, “A graph-based taxonomy of citation recommendation models,” *Artificial Intelligence Review*, vol. 53, pp. 5217–5260, 2020. [5](#)
- [35] M. J. Zaki, M. Peters, I. Assent, and T. Seidl, “Clicks: An effective algorithm for mining subspace clusters in categorical datasets,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 736–742, 2005. [6](#), [32](#)
- [36] A. Bärmann, P. Gemander, M. Merkert, A.-K. Wiertz, and F. J. Z. Martínez, “Algorithms for the clique problem with multiple-choice constraints under a series-parallel dependency graph,” *Discrete Applied Mathematics*, vol. 324, pp. 145–166, 2023. [6](#), [32](#)
- [37] G. Ouyang, D. K. Dey, and P. Zhang, “Clique-based method for social network clustering,” *Journal of Classification*, vol. 37, no. 1, pp. 254–274, 2020. [6](#)
- [38] H. Yu, A. Paccanaro, V. Trifonov, and M. Gerstein, “Predicting interactions in protein networks by completing defective cliques,” *Bioinformatics*, vol. 22, no. 7, pp. 823–829, 2006. [6](#)
- [39] K. Rozman, A. Ghysels, B. Zavalnij, T. Kunej, U. Bren, D. Janežič, and J. Konc, “Enhanced molecular docking: Novel algorithm for identifying highest weight k-cliques in weighted general and protein-ligand graphs,” *Journal of molecular structure*, vol. 1304, p. 137639, 2024. [6](#)
- [40] M. C. Schmidt, N. F. Samatova, K. Thomas, and B.-H. Park, “A scalable, parallel algorithm for maximal clique enumeration,” *Journal of parallel and distributed computing*, vol. 69, no. 4, pp. 417–428, 2009. [6](#), [39](#), [75](#)

- [41] C. A. Phillips, K. Wang, E. J. Baker, J. A. Bubier, E. J. Chesler, and M. A. Langston, “On finding and enumerating maximal and maximum k-partite cliques in k-partite graphs,” *Algorithms*, vol. 12, no. 1, p. 23, 2019. [6](#), [12](#), [14](#), [34](#), [39](#)
- [42] C. Bron and J. Kerbosch, “Algorithm 457: finding all cliques of an undirected graph,” *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973. [6](#), [34](#)
- [43] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, “Parallel maximum clique algorithms with applications to network analysis,” *SIAM Journal on Scientific Computing*, vol. 37, no. 5, pp. C589–C616, 2015. [6](#), [75](#)
- [44] P. San Segundo, D. Rodríguez-Losada, and A. Jiménez, “An exact bit-parallel algorithm for the maximum clique problem,” *Computers & Operations Research*, vol. 38, no. 2, pp. 571–581, 2011. [6](#), [75](#)
- [45] K. Fuchimoto, T. Ishii, and M. Ueno, “Hybrid maximum clique algorithm using parallel integer programming for uniform test assembly,” *IEEE Transactions on Learning Technologies*, vol. 15, no. 2, pp. 252–264, 2022. [6](#), [75](#)
- [46] E. Pelofske, G. Hahn, and H. N. Djidjev, “Solving larger maximum clique problems using parallel quantum annealing,” *Quantum Information Processing*, vol. 22, no. 5, p. 219, 2023. [6](#), [75](#)
- [47] L. Chang, “Efficient maximum clique computation and enumeration over large sparse graphs,” *The VLDB Journal*, vol. 29, no. 5, pp. 999–1022, 2020. [6](#)
- [48] C. Bleker, S. K. Grady, and M. A. Langston, “A comparative study of gene co-expression thresholding algorithms,” *Journal of Computational Biology*, vol. 31, no. 6, pp. 539–548, 2024. [7](#), [58](#), [60](#), [65](#), [67](#)
- [49] S. S. Ou-Yang, J. Y. Lu, X. Q. Kong, Z. J. Liang, C. Luo, and H. Jiang, “Computational drug discovery,” *Acta Pharmacol Sin*, vol. 33, no. 9, pp. 1131–40, 2012. [11](#)

- [50] J. Zhao, Y. Cao, and L. Zhang, “Exploring the computational methods for protein-ligand binding site prediction,” *Comput Struct Biotechnol J*, vol. 18, pp. 417–426, 2020. [11](#), [12](#)
- [51] C. M. Song, S. J. Lim, and J. C. Tong, “Recent advances in computer-aided drug design,” *Brief Bioinform*, vol. 10, no. 5, pp. 579–91, 2009. [11](#)
- [52] S. Pushpakom, F. Iorio, P. A. Eyers, K. J. Escott, S. Hopper, A. Wells, A. Doig, T. Guilleams, J. Latimer, C. McNamee, A. Norris, P. Sanseau, D. Cavalla, and M. Pirmohamed, “Drug repurposing: progress, challenges and recommendations,” *Nat Rev Drug Discov*, vol. 18, no. 1, pp. 41–58, 2019. [11](#)
- [53] T. J. Rinsema, “One hundred years of aspirin,” *Med Hist*, vol. 43, no. 4, pp. 502–7, 1999. [11](#)
- [54] J. P. Jourdan, R. Bureau, C. Rochais, and P. Dallemagne, “Drug repositioning: a brief overview,” *J Pharm Pharmacol*, vol. 72, no. 9, pp. 1145–1151, 2020. [11](#)
- [55] P. M. P. Rothwell, F. G. R. P. Fowkes, J. F. F. P. Belch, H. M. D. Ogawa, C. P. P. Warlow, and T. W. P. Meade, “Effect of daily aspirin on long-term risk of death due to cancer: analysis of individual patient data from randomised trials,” *The Lancet (British edition)*, vol. 377, no. 9759, pp. 31–41, 2011. [11](#)
- [56] S. C, D. K. S, V. Ragunathan, P. Tiwari, S. A, and B. D. P, “Molecular docking, validation, dynamics simulations, and pharmacokinetic prediction of natural compounds against the sars-cov-2 main-protease,” *J Biomol Struct Dyn*, vol. 40, no. 2, pp. 585–611, 2022. [12](#)
- [57] J. Liu, Z. Zuo, and G. Wu, “Link prediction only with interaction data and its application on drug repositioning,” *IEEE Trans Nanobioscience*, vol. 19, no. 3, pp. 547–555, 2020. [12](#)
- [58] E. Gündoğan and B. Kaya, “A recommendation method based on link prediction in drug-disease bipartite network,” in *2017 2nd International*

Conference on Advanced Information and Communication Technologies (AICT), pp. 125–128, IEEE, 2017. [12](#)

- [59] W. Wang, H. Lv, Y. Zhao, D. Liu, Y. Wang, and Y. Zhang, “Dls: A link prediction method based on network local structure for predicting drug-protein interactions,” *Front Bioeng Biotechnol*, vol. 8, p. 330, 2020. [12](#)
- [60] Y. Luo, X. Zhao, J. Zhou, J. Yang, Y. Zhang, W. Kuang, J. Peng, L. Chen, and J. Zeng, “A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information,” *Nat Commun*, vol. 8, no. 1, p. 573, 2017. [12](#)
- [61] F. MacLean, “Knowledge graphs and their applications in drug discovery,” *Expert Opinion on Drug Discovery*, vol. 16, no. 9, pp. 1057–1069, 2021. [12](#)
- [62] H. Eslami Manoochehri and M. Nourani, “Drug-target interaction prediction using semi-bipartite graph model and deep learning,” *BMC Bioinformatics*, vol. 21, no. 4, p. 248, 2020. [12](#)
- [63] F. Cheng, C. Liu, J. Jiang, W. Lu, W. Li, G. Liu, W. Zhou, J. Huang, and Y. Tang, “Prediction of drug-target interactions and drug repositioning via network-based inference,” *PLoS Comput Biol*, vol. 8, no. 5, p. e1002503, 2012. [12](#)
- [64] M. Zitnik, R. Sosič, S. Maheshwari, and J. Leskovec, “BioSNAP Datasets: Stanford biomedical network dataset collection.” <http://snap.stanford.edu/biodata>, aug 2018. [13](#), [52](#)
- [65] D. S. Wishart, Y. D. Feunang, A. C. Guo, E. J. Lo, A. Marcu, J. R. Grant, T. Sajed, D. Johnson, C. Li, Z. Sayeeda, N. Assempour, I. Iynkkaran, Y. Liu, A. Maciejewski, N. Gale, A. Wilson, L. Chin, R. Cummings, D. Le, A. Pon, C. Knox, and M. Wilson, “Drugbank 5.0: a major update to the drugbank

database for 2018,” *Nucleic Acids Res*, vol. 46, no. D1, pp. D1074–D1082, 2018.

13

- [66] S. S. Group, “Miner: Gigascale multimodal biological network.” 2017. 13
- [67] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegiers, T. C. Wiegiers, and C. J. Mattingly, “The comparative toxicogenomics database: update 2017,” *Nucleic Acids Res*, vol. 45, no. D1, pp. D972–D978, 2017. 13
- [68] J. S. Amberger, C. A. Bocchini, A. F. Scott, and A. Hamosh, “Omim.org: leveraging knowledge across phenotype-gene relationships,” *Nucleic Acids Res*, vol. 47, no. D1, pp. D1038–D1043, 2019. 13
- [69] J. A. Bondy, U. S. R. Murty, *et al.*, *Graph theory with applications*, vol. 290. Macmillan London, 1976. 13
- [70] S. J. Nelson, M. Schopen, A. G. Savage, J.-L. Schulman, and N. Arluk, “The mesh translation maintenance system: structure, interface design, and implementation,” in *MEDINFO 2004*, pp. 67–69, IOS Press, 2004. 14
- [71] R. D. Hagan, M. A. Langston, and K. Wang, “Lower bounds on paraclique density,” *Discrete Applied Mathematics*, vol. 204, pp. 208–212, 2016. 15
- [72] D. Bajusz, A. Rácz, and K. Héberger, “Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?,” *Journal of Cheminformatics*, vol. 7, no. 1, p. 20, 2015. 18
- [73] G. Landrum, “Rdkit: Open-source cheminformatics software, 2016,” *URL* <http://www.rdkit.org/>, <https://github.com/rdkit/rdkit>, vol. 149, p. 150, 2016. 18

- [74] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, “Reoptimization of mdl keys for use in drug discovery,” *Journal of Chemical Information and Computer Sciences*, vol. 42, no. 6, pp. 1273–1280, 2002. [18](#)
- [75] H. Kuwahara and X. Gao, “Analysis of the effects of related fingerprints on molecular similarity using an eigenvalue entropy approach,” *Journal of cheminformatics*, vol. 13, pp. 1–12, 2021. [18](#)
- [76] M. Vogt and J. Bajorath, “ccbmllib—a python package for modeling tanimoto similarity value distributions,” *F1000Research*, vol. 9, pp. Chem–Inf, 2020. [18](#), [96](#)
- [77] G. Bindea, B. Mlecnik, H. Hackl, P. Charoentong, M. Tosolini, A. Kirilovsky, W. H. Fridman, F. Pages, Z. Trajanoski, and J. Galon, “Cluego: a cytoscape plug-in to decipher functionally grouped gene ontology and pathway annotation networks,” *Bioinformatics*, vol. 25, no. 8, pp. 1091–3, 2009. [21](#)
- [78] D. Yang, Q. Zhou, V. Labroska, S. Qin, S. Darbalaei, Y. Wu, E. Yuliantie, L. Xie, H. Tao, J. Cheng, Q. Liu, S. Zhao, W. Shui, Y. Jiang, and M. W. Wang, “G protein-coupled receptors: structure- and function-based drug discovery,” *Signal Transduct Target Ther*, vol. 6, no. 1, p. 7, 2021. [21](#)
- [79] K. Sriram and P. A. Insel, “G protein-coupled receptors as targets for approved drugs: how many targets and how many drugs?,” *Molecular pharmacology*, vol. 93, no. 4, pp. 251–258, 2018. [21](#)
- [80] Y. Ji, M. Karbaschi, and M. S. Cooke, “Mycoplasma infection of cultured cells induces oxidative stress and attenuates cellular base excision repair activity,” *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, vol. 845, p. 403054, 2019. [25](#)
- [81] A. W. Boots, M. Drent, E. L. Swennen, H. J. Moonen, A. Bast, and G. R. Haenen, “Antioxidant status associated with inflammation in sarcoidosis: a

- potential role for antioxidants,” *Respiratory medicine*, vol. 103, no. 3, pp. 364–372, 2009. [25](#)
- [82] G. Di Maio, I. Villano, C. R. Ilardi, A. Messina, V. Monda, A. C. Iodice, C. Porro, M. A. Panaro, S. Chieffi, G. Messina, M. Monda, and M. La Marra, “Mechanisms of transmission and processing of pain: A narrative review,” *Int J Environ Res Public Health*, vol. 20, no. 4, 2023. [25](#)
- [83] J. D. Hutcheson, V. Setola, B. L. Roth, and W. D. Merryman, “Serotonin receptors and heart valve disease—it was meant 2b,” *Pharmacology & therapeutics*, vol. 132, no. 2, pp. 146–157, 2011. [25](#)
- [84] H. N. Pawar, S. Balivada, and M. J. Kenney, “Does acute heat stress differentially-modulate expression of ionotropic neurotransmitter receptors in the rvlm of young and aged f344 rats?,” *Neuroscience letters*, vol. 687, pp. 223–233, 2018. [25](#)
- [85] S. Majumdar, “Role of glutamate in the development of visual pathways,” *Frontiers in Ophthalmology*, vol. 3, p. 1147769, 2023. [25](#)
- [86] K. S. Dickson, D. L. Rosenstein, and N. A. Sowa, “Recurrent anti-nmda receptor encephalitis after mycoplasma pneumonia infection,” *American Journal of Psychiatry*, vol. 180, no. 12, pp. 880–883, 2023. [25](#)
- [87] A. A. Gilep, T. A. Sushko, and S. A. Usanov, “At the crossroads of steroid hormone biosynthesis: the role, substrate specificity and evolutionary development of cyp17,” *Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics*, vol. 1814, no. 1, pp. 200–209, 2011. [25](#)
- [88] O. Ayaz and S. E. Howlett, “Testosterone modulates cardiac contraction and calcium homeostasis: cellular and molecular mechanisms,” *Biology of sex differences*, vol. 6, pp. 1–15, 2015. [25](#)

- [89] L. Li, J. Wu, M. Luo, Y. Sun, and G. Wang, "The effect of heat stress on gene expression, synthesis of steroids, and apoptosis in bovine granulosa cells," *Cell Stress and Chaperones*, vol. 21, no. 3, pp. 467–475, 2016. [25](#)
- [90] O. Trott and A. J. Olson, "Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *J Comput Chem*, vol. 31, no. 2, pp. 455–61, 2010. [26](#)
- [91] C. C. G. ULC, "Molecular operating environment (moe), 2019.01," 2020. [26](#)
- [92] S. R. Ellingson, J. C. Smith, and J. Baudry, "Vinampi: facilitating multiple receptor high-throughput virtual docking on high-performance computers," *J Comput Chem*, vol. 34, no. 25, pp. 2212–21, 2013. [28](#)
- [93] N. M. O’Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, "Open babel: An open chemical toolbox," *J Cheminform*, vol. 3, p. 33, 2011. [28](#)
- [94] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson, "Autodock4 and autodocktools4: Automated docking with selective receptor flexibility," *J Comput Chem*, vol. 30, no. 16, pp. 2785–91, 2009. [28](#)
- [95] S. K. Burley, C. Bhikadiya, C. Bi, S. Bittrich, L. Chen, G. V. Crichlow, C. H. Christie, K. Dalenberg, L. Di Costanzo, J. M. Duarte, S. Dutta, Z. Feng, S. Ganesan, D. S. Goodsell, S. Ghosh, R. K. Green, V. Guranovic, D. Guzenko, B. P. Hudson, C. L. Lawson, Y. Liang, R. Lowe, H. Namkoong, E. Peisach, I. Persikova, C. Randle, A. Rose, Y. Rose, A. Sali, J. Segura, M. Sekharan, C. Shao, Y. P. Tao, M. Voigt, J. D. Westbrook, J. Y. Young, C. Zardecki, and M. Zhuravleva, "Rcsb protein data bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering

- and energy sciences,” *Nucleic Acids Res*, vol. 49, no. D1, pp. D437–D451, 2021. [28](#)
- [96] W. Humphrey, A. Dalke, and K. Schulten, “Vmd: visual molecular dynamics,” *J Mol Graph*, vol. 14, no. 1, pp. 33–8, 27–8, 1996. [28](#)
- [97] S. Szabó and B. Zaválnij, “Clique search in graphs of special class and job shop scheduling,” *Mathematics*, vol. 10, no. 5, p. 697, 2022. [32](#), [34](#)
- [98] C. A. Phillips, K. Wang, J. Bubier, E. J. Baker, E. J. Chesler, and M. A. Langston, “Scalable multipartite subgraph enumeration for integrative analysis of heterogeneous experimental functional genomics data,” in *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pp. 626–633, 2015. [32](#)
- [99] M. Mirghorbani and P. Krokhmal, “On finding k-cliques in k-partite graphs,” *Optimization Letters*, vol. 7, pp. 1155–1165, 2013. [34](#)
- [100] C. Jeong, S. Jang, E. Park, and S. Choi, “A context-aware citation recommendation model with bert and graph convolutional networks,” *Scientometrics*, vol. 124, pp. 1907–1922, 2020. [58](#)
- [101] B. Van de Sande, C. Flerin, K. Davie, M. De Waegeneer, G. Hulselmans, S. Aibar, R. Seurinck, W. Saelens, R. Cannoodt, Q. Rouchon, *et al.*, “A scalable scenic workflow for single-cell gene regulatory network analysis,” *Nature protocols*, vol. 15, no. 7, pp. 2247–2276, 2020. [58](#)
- [102] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, *et al.*, “Towards a proteome-scale map of the human protein–protein interaction network,” *Nature*, vol. 437, no. 7062, pp. 1173–1178, 2005. [58](#)
- [103] D. F. Burke, P. Bryant, I. Barrio-Hernandez, D. Memon, G. Pozzati, A. Shenoy, W. Zhu, A. S. Dunham, P. Albanese, A. Keller, *et al.*, “Towards a structurally

- resolved human protein interaction network,” *Nature Structural & Molecular Biology*, vol. 30, no. 2, pp. 216–225, 2023. 58
- [104] L. Qiao, L. Zhang, S. Chen, and D. Shen, “Data-driven graph construction and graph learning: A review,” *Neurocomputing*, vol. 312, pp. 336–351, 2018. 58
- [105] S. Lee, “Mastering proximity graphs in computational geometry.” <https://www.numberanalytics.com/blog/mastering-proximity-graphs-computational-geometry>, 2025. 59
- [106] K. R. Gabriel and R. R. Sokal, “A new statistical approach to geographic variation analysis,” *Systematic zoology*, vol. 18, no. 3, pp. 259–278, 1969. 60
- [107] M. Aupetit, “High-dimensional labeled data analysis with gabriel graphs,” in *ESANN*, pp. 21–26, 2003. 60
- [108] G. Oliva, S. Panzieri, F. Pascucci, and R. Setola, “Sensor networks localization: Extending trilateration via shadow edges,” *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2752–2755, 2015. 60
- [109] M. J. Alenazi and J. P. Sterbenz, “Comprehensive comparison and accuracy of graph metrics in predicting network resilience,” in *2015 11th international conference on the design of reliable communication networks (DRCN)*, pp. 157–164, IEEE, 2015. 60
- [110] A. Diaz, S. J. Liu, C. Sandoval, A. Pollen, T. J. Nowakowski, D. A. Lim, and A. Kriegstein, “Scell: integrated analysis of single-cell rna-seq data,” *Bioinformatics*, vol. 32, no. 14, pp. 2219–2220, 2016. 60
- [111] T.-H. Su and R.-C. Chang, “The k-gabriel graphs and their applications,” in *International Symposium on Algorithms*, pp. 66–75, Springer, 1990. 60

- [112] T. Kaiser, M. Saumell, and N. Van Cleemput, “10-gabriel graphs are hamiltonian,” *Information Processing Letters*, vol. 115, no. 11, pp. 877–881, 2015. [60](#)
- [113] T. Barrett, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, M. Holko, *et al.*, “Ncbi geo: archive for functional genomics data sets—update,” *Nucleic acids research*, vol. 41, no. D1, pp. D991–D995, 2012. [64](#)
- [114] S. Davis and P. S. Meltzer, “Geoquery: a bridge between the gene expression omnibus (geo) and bioconductor,” *Bioinformatics*, vol. 23, no. 14, pp. 1846–1847, 2007. [64](#)
- [115] A. D. Perkins and M. A. Langston, “Threshold selection in gene co-expression networks using spectral graph theory techniques,” in *BMC bioinformatics*, vol. 10, pp. 1–11, Springer, 2009. [65](#)
- [116] E. J. Chesler and M. A. Langston, “Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data,” in *RECOMB Workshop on Systems Biology*, pp. 150–165, Springer, 2005. [65](#)
- [117] R. D. Hagan, M. A. Langston, and K. Wang, “Lower bounds on paraclique density,” *Discrete Applied Mathematics*, vol. 204, pp. 208–212, 2016. [67](#)
- [118] Z. Fang, X. Liu, and G. Peltz, “Gseapy: a comprehensive package for performing gene set enrichment analysis in python,” *Bioinformatics*, vol. 39, no. 1, p. btac757, 2023. [67](#)
- [119] K. Abbas, A. Abbasi, S. Dong, L. Niu, L. Yu, B. Chen, S.-M. Cai, and Q. Hasan, “Application of network link prediction in drug discovery,” *BMC bioinformatics*, vol. 22, pp. 1–21, 2021. [74](#)
- [120] T. Gaudet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. Hayter, R. Vickers, C. Roberts, J. Tang, *et al.*, “Utilizing graph machine learning within

- drug discovery and development,” *Briefings in bioinformatics*, vol. 22, no. 6, p. bbab159, 2021. [74](#)
- [121] C. Zhang, T. Zang, and T. Zhao, “Kge-unit: toward the unification of molecular interactions prediction based on knowledge graph and multi-task learning on drug discovery,” *Briefings in Bioinformatics*, vol. 25, no. 2, p. bbae043, 2024. [74](#)
- [122] F. T. Mukta, M. M. Rana, A. Meyer, S. Ellingson, and D. D. Nguyen, “The algebraic extended atom-type graph-based model for precise ligand–receptor binding affinity prediction,” *Journal of Cheminformatics*, vol. 17, no. 1, p. 10, 2025. [74](#)
- [123] A. Das, S.-V. Sanei-Mehri, and S. Tirthapura, “Shared-memory parallel maximal clique enumeration from static and dynamic graphs,” *ACM Transactions on Parallel Computing (TOPC)*, vol. 7, no. 1, pp. 1–28, 2020. [75](#)
- [124] Q. Chen, C. Fang, Z. Wang, B. Suo, Z. Li, and Z. G. Ives, “Parallelizing maximal clique enumeration over graph data,” in *Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part II 21*, pp. 249–264, Springer, 2016. [75](#)
- [125] M. Almasri, I. E. Hajj, R. Nagi, J. Xiong, and W.-m. Hwu, “Parallel k-clique counting on gpus,” in *Proceedings of the 36th ACM International Conference on Supercomputing*, pp. 1–14, 2022. [75](#)
- [126] J. Khalil, D. Yan, G. Guo, and L. Yuan, “Parallel mining of large maximal quasi-cliques,” *The VLDB Journal*, vol. 31, no. 4, pp. 649–674, 2022. [75](#)
- [127] Y.-W. Wei, W.-M. Chen, and H.-H. Tsai, “Accelerating the bron-kerbosch algorithm for maximal clique enumeration using gpus,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2352–2366, 2021. [76](#)

- [128] Y. Yang, F. Hao, B. Pang, G. Min, and Y. Wu, “Dynamic maximal cliques detection and evolution management in social internet of things: A formal concept analysis approach,” *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1020–1032, 2022. [76](#)
- [129] S. Khoshraftar, A. An, and N. Babanejad, “Temporal graph representation learning via maximal cliques,” in *2022 IEEE International Conference on Big Data (Big Data)*, pp. 606–615, 2022. [76](#)
- [130] M. Lahby, A. Essouiri, and A. Sekkaki, “A novel modeling approach for vertical handover based on dynamic k-partite graph in heterogeneous networks,” *Digital Communications and Networks*, vol. 5, no. 4, pp. 297–307, 2019. [76](#)
- [131] J. Cohen, *Statistical power analysis for the behavioral sciences*. routledge, 2013. [96](#)
- [132] C. M. Weisman, A. W. Murray, and S. R. Eddy, “Many, but not all, lineage-specific genes can be explained by homology detection failure,” *PLoS biology*, vol. 18, no. 11, p. e3000862, 2020. [97](#)
- [133] A. Friedrich, R. Ripp, N. Garnier, E. Bettler, G. Deléage, O. Poch, and L. Moulinier, “Blast sampling for structural and functional analyses,” *BMC bioinformatics*, vol. 8, pp. 1–17, 2007. [97](#)
- [134] B. Buchfink, C. Xie, and D. H. Huson, “Fast and sensitive protein alignment using diamond,” *Nature methods*, vol. 12, no. 1, pp. 59–60, 2015. [97](#)
- [135] S. Vilar, G. Cozza, and S. Moro, “Medicinal chemistry and the molecular operating environment (moe): application of qsar and molecular docking to drug discovery,” *Curr Top Med Chem*, vol. 8, no. 18, pp. 1555–72, 2008. [100](#)

Appendix A

Supplementary Analysis of Chapter 2 Experiments

A.1 Extended Analysis for Section 2.3

In this appendix, we provide an extended analysis of the results presented in Section 2.3 and offer additional insights. For an imputed edge e between disease x and drug y , we created a set $D_1(e)$ of Tanimoto similarity scores between the fingerprint of y and the fingerprints of all drugs found in tricliques that produced this same edge during triclique augmentation. We did this to ensure that y and each of these drugs were adjacent to at least one common protein and one common disease. Next, we created a set $D_2(e)$ of Tanimoto similarity scores between y and drugs not adjacent to x via known or imputed edges. Our reasoning for these computations was that higher D_1 scores relative to D_2 scores may have provided evidence that this imputed edge was valid. From there, we constructed U_1 as the union of all $D_1(e)$ sets, and U_2 as the union of all $D_2(e)$ sets, where e ranged over all imputed disease-drug edges. Finally, we compared the distributions of U_1 and U_2 using the non-parametric one-sided Mann-Whitney U test, after Shapiro-Wilk tests confirmed that U_1 and U_2 did

not follow normal distributions. The Shapiro-Wilk tests yielded p -values of $2.24e^{-46}$ and $6.10e^{-73}$ for U_1 and U_2 , respectively.

We used the procedure just described to collect 32,310 available Tanimoto drug similarity scores in U_1 and 434,302 scores in U_2 , with means of 0.332 and 0.303, respectively. Under the hypothesis that the scores of U_1 would be higher than U_2 , we obtained a p -value of $5.00e^{-280}$ from the one-sided Mann-Whitney U test, much less than our confidence cutoff of 0.05. The Cohen’s d effect size for this test was 0.219, which is typically considered a small effect (131). This result supported our expectation that U_1 would have higher similarity scores on average than U_2 , although the difference was modest. We also used ccbmlib (76) to estimate the p -values of the Tanimoto similarity scores themselves. We filtered both U_1 and U_2 to remove Tanimoto similarity scores with p -values not less than 0.05. After filtering, U_1 retained 2,768 scores (8.57%) and U_2 retained 23,369 scores (5.38%). We applied the one-sided Mann-Whitney U test to these two subsets, producing a p -value of $4.50e^{-9}$, and a Cohen’s d effect size of 0.152. These differences were again modest but statistically significant.

A.2 Extended Analysis for Section 2.4

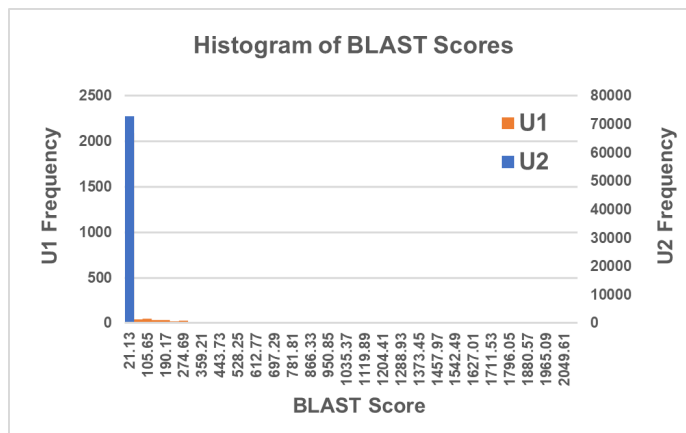
We constructed new sets U_1 and U_2 following the steps outlined in Appendix A.1, using BLAST bit scores of protein sequences in the place of Tanimoto similarity scores of drug fingerprints. U_1 comprised 2,444 BLAST bit scores, while U_2 comprised 72,919 BLAST bit scores, with mean values of 59.06 and 14.54, respectively. Figure A.1a shows the distributions of BLAST bit scores in U_1 and U_2 . We conducted a one-sided Mann-Whitney U test to evaluate the hypothesis that U_1 had higher scores than U_2 . The resulting p -value of $1.63e^{-88}$ was significantly lower than our cutoff of 0.05, supporting our hypothesis. Next, we filtered these sets based on a specified E-value and bit score threshold, respectively. While there is no standard recognized cutoff for determining BLAST alignment significance, some consider E-values less than

0.001 (132; 133) or bit scores exceeding 50 (134) to indicate statistically meaningful similarity. In U_1 , 438(17.9%) BLAST results exhibited an E-value less than 0.001, while 7906 (10.8%) results in U_2 had E-values less than 0.001 (Figure A.1b). We applied the one-sided Mann-Whitney U test exclusively to these subsets of U_1 and U_2 , yielding a p -value of approximately 0. In U_1 , 342 (14.0%) values had bit scores greater than 50, while only 143 (0.2%) scores in U_2 exceeded this threshold (Figure A.1c). We applied the one-sided Mann-Whitney U test exclusively to these subsets of U_1 and U_2 , yielding a p -value of $1.40e^{-5}$. Notably, all BLAST alignments in our analysis with bit scores greater than 50 had E-values below 0.001. Specifically, among the BLAST alignments in U_1 with E-values less than 0.001, 78.1% also had bit scores greater than 50, in stark contrast to only 1.8% in U_2 .

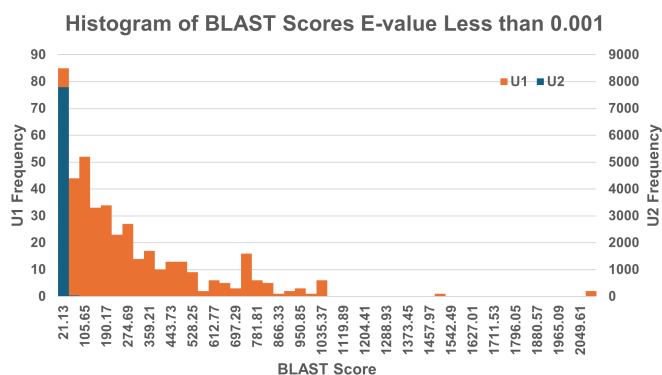
A.3 Molecular Docking Protocol Validation

To evaluate the global search docking protocol, we compared multiple docking results with crystal structures from the RCSB Protein Data Bank (PDB) for a protein complexed with one of the known binding drugs in our dataset. We selected multiple docking results from our preliminary study, in which we applied triclique augmentation to a small set of triclques with the lowest possible nonadjacency bound value of one. We carried out an initial investigation prior to performing comprehensive studies such as nonadjacency bound selection, by applying triclique augmentation to 40 triclques, all with a nonadjacency bound of one. We selected these triclques by sorting all triclques into four sets: three sets based on the size of each partite set and one set based on triclique size. We then selected the ten largest triclques from each set.

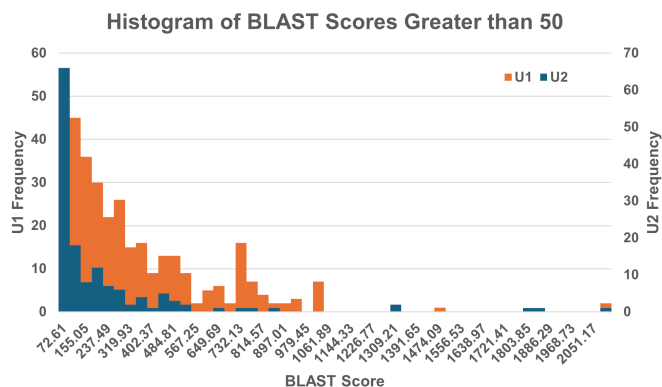
We identified 45 proteins in the imputed drug–protein edges in the preliminary study and used them to search for crystal structures of complexes with one of the known binding drugs in our dataset. Among these, we identified eight proteins with the required crystal structures available in the PDB. For five of the eight proteins



(a) Histogram of BLAST scores



(b) Histogram of BLAST scores e-value Less than 0.001



(c) Histogram of BLAST Scores Greater than 50

Figure A.1: Histograms of the BLAST bit score distributions. The yellow histogram represents the BLAST bit scores from U_1 and is measured using the left y -axis. The blue histogram shows the BLAST bit scores from U_2 and is measured using the right y -axis. (a) Histogram of all BLAST bit scores. (b) Histogram of BLAST bit scores with an E-value below 0.001. (c) Histogram of BLAST bit scores greater than 50.

Table A.1: Best model RMSD between the docked and co-crystal ligands.

	PDB				
	3TBG	6KUW	7C61	7EJ8	7JVR
Drugbank	DB00679	DB04540	DB00696	DB00484	DB01200
RMSD	1.3787	1.5111	3.9803	2.3087	9.6884
Conformation Number	1	2	1	1	2

(3TBG, 6KUW, 7C61, 7EJ8, and 7JVR), we successfully docked the corresponding drugs into the correct binding site, with the best case of 1.38 Å root mean squared deviation (RMSD) to the crystal structure (Figure A.2). Each drug-protein docking result provided nine predicted binding conformations, which we then ordered by binding energy. For the five drug-protein pairs, we found the best RMSD occurred in either the first or second conformation. Table A.1 contains the best RMSD between the docked model and the co-crystallized compound for these five drug-protein pairs and indicate in which conformation the best RMSD occurred. For the largest RMSD between the docked and co-crystalized ligand in 7JVR, six of the nine models were in the binding site of the crystal structure, but the molecule was buried deeper in the crystal structure than what docking predicted.

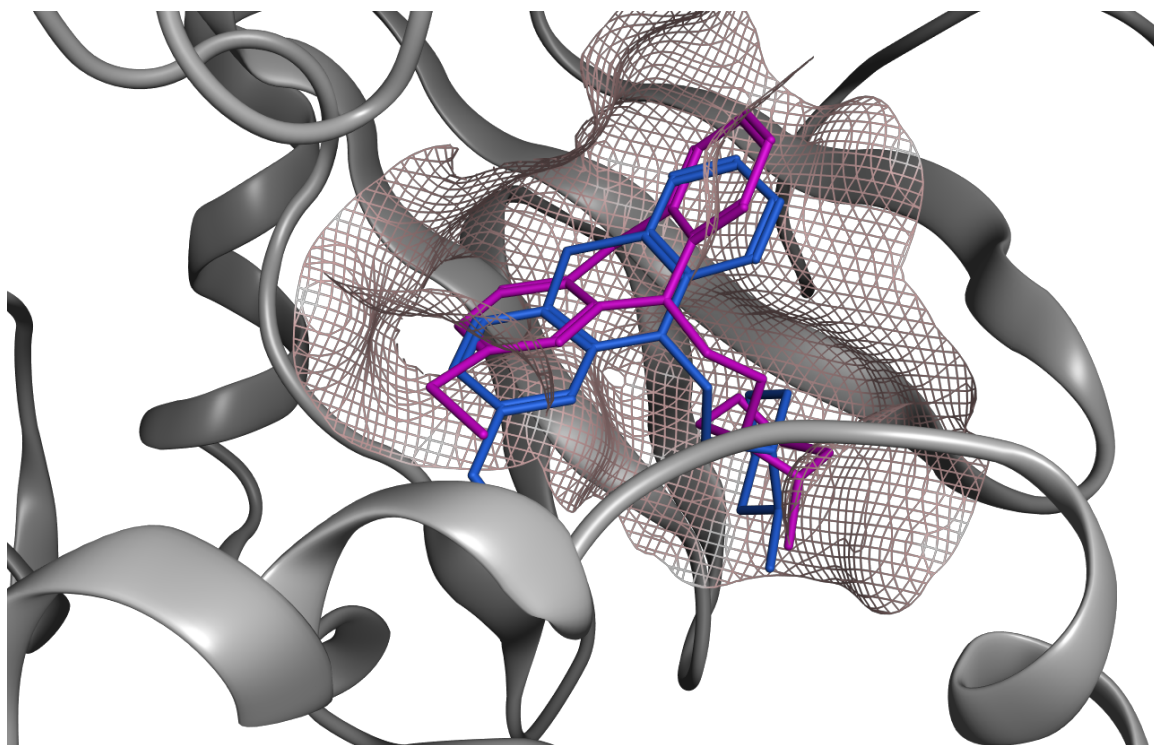


Figure A.2: Comparison of docked (blue) and co-crystallized (magenta) DB00679 from PDB ID 3TBG. The protein backbone is gray and the protein surface within 3.5 Å of the ligand is displayed with lines. A 1.38 Å RMSD between the docked ligand and co-crystallized ligand shows that a global search docking protocol (docking to the entire protein) was able to identify the binding site in Human cytochrome P450 2D6 for Thioridazine correctly. Figure generated in MOE ([135](#))

Appendix B

Datasets for Chapter [4](#)

Experiments

Table B.1: Summary of *homo sapiens* GEO data sets used in graph construction experiments

GEO	Disease	Label	Size
GSE10220	Acute coronary syndrome	Macrophage	48
		Monocyte	48
GSE10245	Lung cancer	Adenocarcinoma	40
		Squamous	18
GSE11501	Celiac disease	Celiac	110
		Healthy	22
GSE19429	Myelodysplastic syndromes	Healthy	17
		MDS	183
GSE20141	Parkinson's disease	Control	8
		Parkinson	10
GSE25507	Autism	Control	64
GSE25837	Chronic loneliness	Lonely	23
		Not lonely	70
GSE30310	HIV infection & ART	HIV-, Fat	7
		HIV-, Muscle	15
		HIV-, PBMC	19
		HIV+ ART, Fat	14
		HIV+ ART, Muscle	30
		HIV+ ART, PBMC	30
		HIV+ No ART, Fat	15
		HIV+ No ART, Muscle	18
		HIV+ No ART, PBMC	18
GSE54514	Sepsis	Control	36
		Sepsis (nonsurvivor)	31
		Sepsis (survivor)	96
GSE8759	Marfan syndrome	Control	41
		Marfan	60
GSE9348	Early stage CRC	Healthy	12
		Tumor	70

Table B.2: Summary of *Mus musculus* GEO datasets used in graph construction experiments

GEO	Comments	Size
GSE15774	Striatum	108
GSE1581	Diverse tissues	87
GSE16496	Diverse tissues of brain	102
GSE22828	Salivary gland	84
GSE24278	Liver of 16 mouse strains	94
GSE24504	Brain, small intestine, kidney, liver, and skeletal muscle	122
GSE24594	Breast cancer	80
GSE2565	Lung	104
GSE26500	Hippocampus and striatum	198
GSE27563	Peripheral blood cell	93
GSE27715	Testicular cell	91
GSE32513	Jejunum, ileum, and colon tissue	144
GSE3327	Seven brain regions in six mouse strains	87
GSE3594	Neural tissues covering the mouse central nervous system	150
GSE4648	C57/BL6 heart tissue	66
GSE6514	Cerebral cortex, hypothalamus	90
GSE97	A diverse array of tissues, organs, and cell lines	90
GSE9954	Different tissues	70

Vita

Cheng Chen was born in China. She received her Bachelor of Science in Bioinformatics from Shanghai Jiao Tong University, followed by a Master of Science in Bioinformatics from the Georgia Institute of Technology. She began her doctoral studies in Computer Science at the University of Tennessee, Knoxville. During her Ph.D., Cheng conducted research in graph algorithms for life sciences data under the supervision of Dr. Michael A. Langston. In addition to her academic work, she completed competitive internships at Meta, Google, and Amazon, contributing to advancements in artificial intelligence, machine learning, software engineering, and data science.