AGH University of Science and Technology
Faculty of Electrical Engineering, Automatics,
Computer Science and Electronics
Department of Computer Science

Exploring complex networks with topological descriptors

Wojciech Czech

Dissertation for the degree of
Doctor of Philosophy

Supervisor: Prof. dr hab. inż. Witold Dzwinel

Kraków, 2012

Akademia Górniczo-Hutnicza
im. Stanisława Staszica
Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Katedra Informatyki

Analiza sieci złożonych za pomocą deskryptorów topologicznych

Wojciech Czech

Praca Doktorska

Promotor: Prof. dr hab. inż. Witold Dzwinel

Kraków, 2012

Abstract

Graph data pervade many fields of contemporary science, being the subject of structural learning and analysis. In recent years we observe increasing volume of structured datasets derived both from complex networks and structural pattern recognition domains. This calls for efficient and general tools capable of measuring similarity of large graphs and perform pattern recognition tasks on structured data.

In this thesis we investigate the problem of graph comparison understood as a construction of robust graph similarity/dissimilarity measure. This allows for moving from structural pattern recognition to statistical pattern recognition field, which is more tackleable from algorithmic perspective. Particularly, we focus on explicit embedding of graphs that is extraction of graph features using topological descriptors. We present a new, general approach to graph embedding based on invariants of distance k-graphs of a given graph. As demonstrated in experimental section, the new framework for graph comparison, can perform better in terms of classification and clusterization accuracy than state-of-art spectral methods. Additionally, after narrowing study to distance k-graphs degree distributions forming so called B-matrix, we got information-rich representation, which constitutes good basis for extracting graph features and visual graph comparison. Moreover, B-matrices are more computationally efficient than currently used $\mathcal{O}(n^3)$ algebraic methods. We perform tests showing that presented approach can be used for analysis of complex networks and structural representations of images.

We also address the graph comparison task from technical perspective by implementing *Graph Investigator* application being a software tool for analysis of groups of graphs. The aim of building this program was to provide interactive framework for comparison of normal and tumor vascular networks. Nevertheless, it can be used for any type of structured data providing numerous topological descriptors together with unsupervised learning and visualization algorithms. In order to improve *Graph Investigator* performance, we used GPU-enabled, optimized implementations of graph algorithms including all-pair shortest-paths R-Kleene algorithm and breadth-first-search. This allows us to analyze large graphs interactively within a 4 seconds time slot.

Lifelessness is merely an appearance, behind which unfamiliar forms of life lie concealed. The range of these forms is endless, their shades and nuances inexhaustible...

Bruno Schulz

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor Dr. Witold Dzwinel (AGH University of Science and Technology) for his guidance, encouragement and continued support. His scientific passion and depth of knowledge have been a source of inspiration during my research work. I have learned a lot from him and I am truly grateful for offering me so much advice and help.

Special thanks are also given to Professor Dr. David Yuen (University of Minnesota), who invited me to visit University of Minnesota and opened my eyes to new research opportunities. He has become my mentor and taught me how to look at scientific problems from a wider perspective. I greatly appreciate his kindness, interest and constructive comments. The collaboration with his team from Minnesota Supercomputing Institute is a great experience for me.

I must also acknowledge Dr. Tomasz Arodź (Virgnia Commonwealth University, AGH University of Science and Technology) for discussion, useful suggestions and providing data for experiments. I am grateful to Dr. Paweł Topa (AGH University of Science and Technology) for sharing with me new concepts as well as lending his support in understanding the model of angiogenesis.

In particular I would like to thank my parents and Sandra for their understanding and continued moral support, owing to which I could finish my dissertation. Special thanks to my sister Aneta for linguistic revision of my manuscript.

I would also like to express my gratitude to all organizations, which provided me with financial support during four years of studies. Special thanks to the local government of Lesser Poland Voivodeship which funded this dissertation with DOCTUS scholarship. I am also grateful to my university which supported me within InnoGrant programme and provided me doctoral stipends. I acknowledge authorities of Institute of Computer Science who granted me with a stipend for young researchers. This work was also partially financed by the Polish Ministry of Science and Higher Education, Project No. N N519 579338.

Contents

Theses of the Dissertation

1. Distance k-graphs constructed on the basis of vertex-vertex dissimilarity measures of a given graph G form ordered set of G-derived graphs, which allows for generation of isomorphism invariants efficient in clusterization and classification on benchmark structured datasets.

2. Performance boost brought by GPU-enabled implementations of graph embedding algorithms allows for increasing size of graphs analyzed interactively by two orders of magnitude.

Chapter 1

Introduction

Graph theory constitutes a general framework for representation and analysis of relations between objects. Since Euler's historical paper on *Seven Bridges of Königsberg* written in 1736, it developed into mature discipline that pervades many areas of contemporary science and engineering. Today, it encompasses many sub-disciplines whose applications range from physics through biology to economics. Graph structures appear as the intersection of many research fields, therefore, the advance of graph analysis tools is crucial for obtaining novel findings and their practical implications. In Figure 1.1 one of possible partitions of graph theory into basic research directions is depicted. The focus of this work is graph comparison sub-discipline, nevertheless the others mentioned in Figure 1.1 are also considered. Figure 1.2 presents research areas that benefit from graph representations and graph analysis tools provided by computer science. Inter alia it shows *theory of complex networks* as one of the core elements. This theory emerged in the last decade as interdisciplinary research area providing deep insight into topology and dynamics of real-world networks and a common viewpoint for their analysis [64]. The four inner elements of the conceptual diagram shown in Figure 1.2 provide a set of general-purpose tools allowing for quantitative description and classification of networks, regardless of their source. The third diagram, depicted in Figure 1.3 presents more detailed view of graphs utilization in computer science sub-disciplines. Particularly, new perspectives of graph-based representations are emerging in such fields as image vision, image processing or sensor networks [111]. In the second part of the dissertation, an overview of structured data used in disciplines depicted in Figure 1.2 and Figure 1.3 is introduced. Furthermore, we describe the methods of graph analysis and recall challenges faced by researchers in this area.

Extensive use of graph structures stems from the following reasons.

- Describing a system as a set of binary relations or interactions among its elements is convenient for humans. Such a fine-grained bottom-up approach reduces complexity and renders the problem easier to comprehend. Depending on mapping of vertex to real object, the edge can reflect association of any type: similarity, correlation, cause-effect relation, physical link, social influence etc.

- Graph vertices form abstraction layer over objects of different types. Typically single vertex represents single object but it can also be mapped to a set of objects (e.g. *class catch cover diagraphs* [110]).

- Heterogenity of relations can be represented using edge weights.

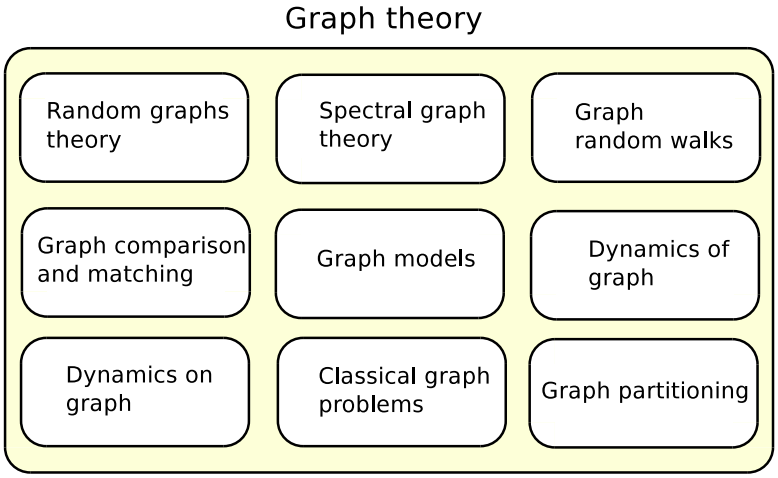


Figure 1.1: Graph theory sub-disciplines. One of many categorization possibilities

- Graphs allow for integration of large amounts of data into one high-level structure capturing system as a whole. This is particularly useful in biomedical or chemical applications where high-throughput experiments produce high volume data that cannot be easily tackled without previous synthesis. In this context, graphs simplify system-level analysis of large ensembles.

In this work we explore graph dissimilarity measures and their applications in learning structural patterns. The motivation for this research is presented in the next section.

1.1 Motivation

With a continuous rise of structured data volume, graph mining becomes complex and computationally challenging task. This is observed both in structural pattern recognition and complex networks areas, where increasing size of analyzed graphs makes currently used algorithms impractical. Examples of such large datasets include graph representations of images [46], metabolic or protein-protein interaction networks [97], transportation networks, Internet and WWW, power grids, protein-folding networks [129], social networks and many others [64, 111]. Even a small sample of WWW network can have the number of vertices of order 10^7, social networks representing interactions between individuals usually have the size of order 10^5. Graph comparison algorithms being the core of graph learning are designed to capture non-trivial, subtle structural differences what usually yields significant time complexity. Although constant growth of computing power allows for increasing size of structured datasets subject to analysis, the development of efficient and robust algorithms is still a challenging task. This is because only a small part of graph measures scale linearly with a graph size. The great part of them rely on shortest-paths or graph random walks which yield time complexity $\mathcal{O}(n^3)$ or even greater.

Measuring graph similarity or dissimilarity allows for application of statistical pattern recognition techniques for objects originally located in the space of graphs which does not possess inbuilt metric. Graph comparison is a crucial research method finding its applications in such tasks involving structured data as confronting models and simulation results with real-world

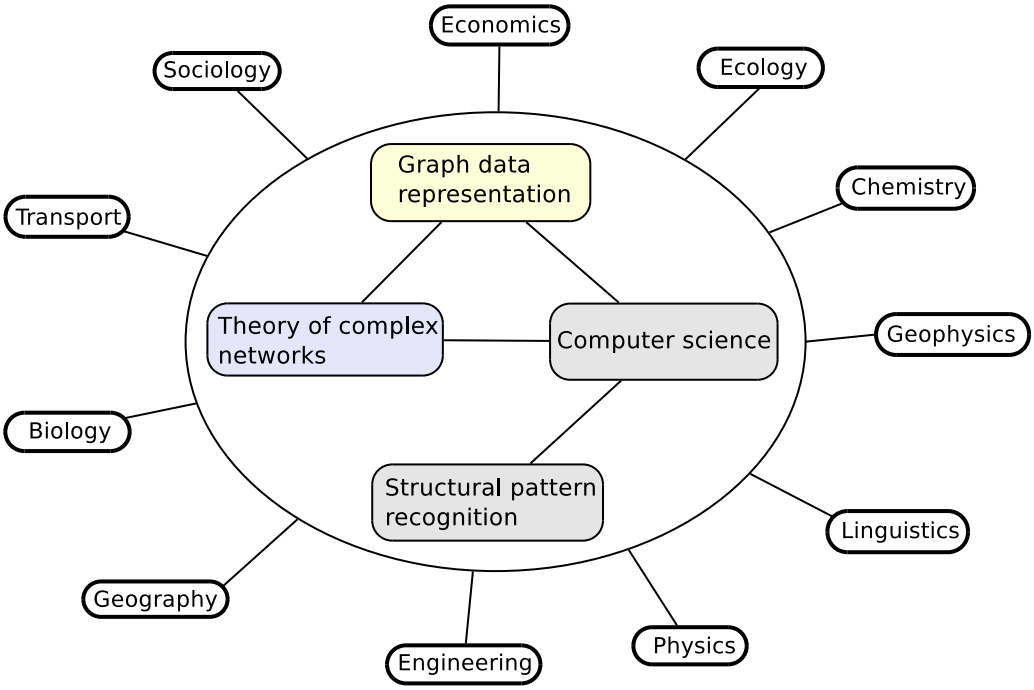


Figure 1.2: Use of graph data representation in various disciplines

data [58], building structured databases, classification and clusterization. The result of graph comparison is dissimilarity or similarity measure that can be used in pattern recognition methods. Due to non-vectorial nature of graphs, their comparison poses some intrinsic problems that cannot be simply neglected during the development of new graph matching algorithms. The direct comparison of two graphs requires enumeration of all sub-substructures and tackling with elements order. The exponential cost of such procedure makes construction of efficient graph metric infeasible. Graph comparison algorithms should also give results invariant under isomorphism, what becomes cumbersome when typical graph representations in a form of adjacency matrices or neighborhood lists are considered.

The practical approach to graph matching uses graph invariants to construct feature vector and embed graph into metric space. The question how to quantitatively capture relevant structural properties of graphs provoked the development of graph measures such as *clustering coefficient*, *efficiency* or *betweenness centrality*, etc. [50]. Today the abundance of scalar graph descriptors makes the selection of relevant features a difficult task that can be tackled, e.g., with a help of information-theoretic tools [32]. Moreover, frequent correlations between typical topological measures influence accuracy of graph classification and clusterization based on pattern vectors [91]. Therefore, the design of general framework for graph invariants generation, which allows for navigation between local and global structural properties and provides information-rich set of features, becomes an important task. This can be approached both from methodological and technical perspective. Firstly, by developing new graph embedding algorithms and secondly by building robust graph analysis software.

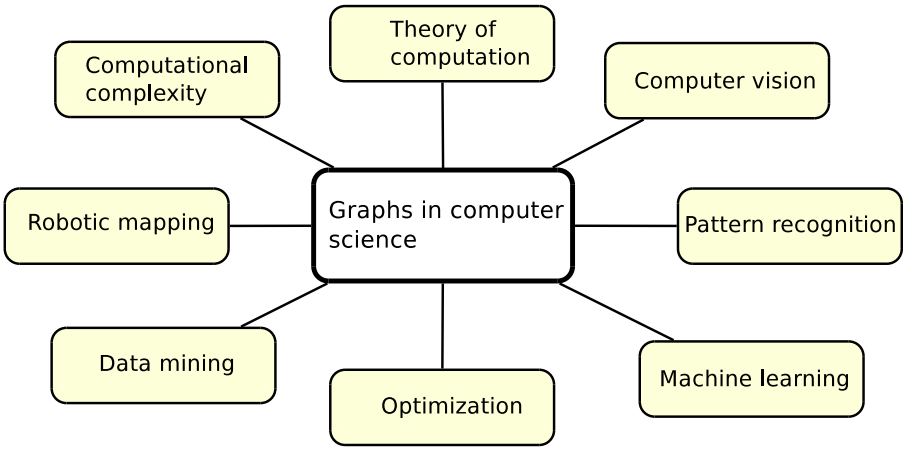


Figure 1.3: Application of graphs in computer science

1.2 Goals and Thesis

The aim of this work is to develop and investigate new, efficient graph comparison methods capable of dealing with large, real-world graphs. To this end we focus on explicit graph embedding algorithms which provide convenient way of dissimilarity-based graph learning. This approach assigns feature vector to a graph yielding linear metric space. The specific goals of research project described in this dissertation are presented below.

- Investigate how to exploit information carried by vertex-vertex metrics for graph embedding and whether this information can be condensed to obtain meaningful, lower-dimensional graph representation.
- Study efficiency of graph descriptors derived from theory of complex networks in structural pattern recognition tasks including image clusterization and classification.
- Build software which allows for extensive, quantitative analysis of structured datasets with the use of graph embedding techniques and statistical pattern recognition algorithms. The focus is put on providing good interactivity and rich set of features, so that the application could serve as ready-to-use, self-contained framework for graph matching.
- Test new graph embedding algorithms on real-world datasets to prove their efficiency. This includes strucutral pattern recognition bechmark datasets and particularly the data obtained from simulations tumor-induced angiogenesis [148, 161].

In order to demonstrate main ideas behind this work and outline author contributions we state following principal theses of this dissertation.

1. **Distance k-graphs constructed on the basis of vertex-vertex dissimilarity measures of a given graph G form ordered set of G-derived graphs, which allows for generation of isomorphism invariants efficient in clusterization and classification on benchmark structured datasets**. Particularly, shortest-path vertex metric applied to build vertex distance k-graphs and edge distance k-graphs yields graph B-matrix representation, which is computationally less expensive than graph invariants

based on Laplace matrix spectral decomposition. Graph feature vectors derived from B-matrices are information rich enough to distinguish graphs with non-trivial structural differences and outperform graph descriptors derived from spectral graph theory. This constitutes general framework for comparison of unweighted and weighted graphs.

2. **Performance boost brought by GPU-enabled implementations of graph embedding algorithms allows for increasing size of graphs analyzed interactively by two orders of magnitude**. Optimized recursive Kleene all-pair shortest-paths algorithm implemented in CUDA can be used to generate vertex B-matrices of 10^4 size graphs in less than 3 seconds (Nvidia Tesla C2070). *Graph Investigator* application provides ready to use framework capable of performing graph clusterization and computation of more than 100 graph descriptors.

1.3 Overview of the Dissertation

The outline of this dissertation is as follows. First, in section 1.4 we introduce notation and recall definitions from graph theory that will be used further in this text. Excluding *Introduction* section, the work is divided into two main parts: *Background* and *Contribution*. The former part describes current state of the art in graph matching including description of sample graph datasets that appear in contemporary science. The latter part presents new findings on graph embedding, describes software built to perform experiments and addresses theses stated in section 1.2.

The *Background* part starts with the chapter presenting approaches to graph analysis with a focus on graph topological descriptors and graph matching algorithms. First, in Section 2.1 we review graph invariants and show how they can be grouped. The more detailed description, including definitions of selected informative graph measures is provided in Appendix A. Next, in Section 2.2 the graph comparison problem is covered. Section 2.2.1 explains types of graph matching and provides general remarks on the topic. Then, in Section 2.2.2 we address exact graph matching problem describing approaches to solving graph isomorphism. The formal definition of graph descriptor is also introduced in this section. The review of currently used graph matching algorithms including pairwise and embedding-based methods is presented in Section 2.2.3. Next, in Chapter 3 the overview of graph data in various research fields is provided. We address applications of structured datasets and describe problems they pose. Particularly, we focus on real-world graphs used in experiments reported in *Contribution* part. This includes graph representations from structural pattern recognition covered in Section 3.2.1, metabolic networks (Section 3.3) and vascular networks described in Section 3.4. The brief review of networks derived from other disciplines is presented in Section 3.5.

In the *Contribution* part, we demonstrate results of our research on graph matching and provide experimental proof of theses stated in Section 1.3. We also describe how the goals of this dissertation were achieved. The contribution of this work is twofold. First, in Chapter 4 which is a theoretical part complemented by experiments, the novel method of graph embedding based on distance k-graphs is presented. Second, Chapters 5 and 6 cover software contributions describing *Graph Investigator* application and its enhancements made with the help of GPU computing. Each chapter in the *Contribution* part ends with *Discussion* section. Finally, Chapter 7 provides dissertation summary with conclusions drawn and some final remarks.

1.4 Notation and definitions

To commence we recall some basic notions from graph theory that will be used further in this text. In this work we consider undirected, unweighted simple graphs primarily, however the ideas for development of presented concepts towards weighted and directed graphs will be also addressed. First, we present definitions of undirected and directed graphs.

Definition 1.4.1 *An undirected graph G is defined as an ordered pair $G = (V(G), E(G))$, where $V(G)$ is a set of vertices and $E(G)$ is a set of edges. An edge $e_{uv} = \{u, v\} \in E(G)$ is an unordered pair of vertices. Two vertices u and v are adjacent ($u \sim v$) if they are joined by an edge.*

Definition 1.4.2 *A directed graph G (digraph) is defined as an ordered pair $G = (V(G), E(G))$, where $V(G)$ is a set of vertices and $E(G)$ is a set of edges. An edge $e_{uv} = (u, v) \in E(G)$ is an ordered pair of vertices.*

Definition 1.4.3 *A walk of length k from vertex u to vertex v is a sequence of k edges connecting u and v. For the closed walk the starting and ending vertex is the same.*

Definition 1.4.4 *An elementary path is a walk without repeating vertices.*

Definition 1.4.5 *The distance between vertex u and v, denoted by $d_G(u, v)$, is a length of the shortest path between u and v. If a path between u and v does not exist then $d_G(u, v) = \infty$.*

Definition 1.4.6 *The diameter of a graph G is a maximal distance between its vertices $diam(G) = \max_{u,v \in V(G)} d_G(u, v)$.*

Definition 1.4.7 *The set of vertices adjacent to vertex v, denoted by N_v, is called its neighborhood. The degree k_v of vertex v is a number of edges which join v with its neighbors.*

Definition 1.4.8 *Graph G is connected if a path exists between each pair of vertices.*

The graph isomorphism is defined as mapping preserving the graph structure represented by its edges. It constitutes an equivalence relation on graphs which partitions the set of all graphs into equivalence classes.

Definition 1.4.9 *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be the graphs without multiple edges. A graph isomorphism between G and H is a bijective mapping: $\alpha : V(G) \longrightarrow V(H)$ such that*

$$(u, v) \in E(G) \Longleftrightarrow (\alpha(u), \alpha(v)) \in E(H).$$

The symbol $G \simeq H$ denotes isomorphic graphs G and H.

Corollary 1.4.10 *If α is graph isomorphism then α^{-1} is also isomorphism.*

Adjacency matrix

Adjacency matrix is a well-known graph representation. Along with adjacency list, it is commonly used in implementations of classic graph-theoretical algorithms as breadth-first traversal, depth-first traversal, minimum spanning tree and maximum flow.

Definition 1.4.11 *Let $G = (V(G), E(G))$ be a graph. Adjacency matrix A_G is a matrix for which*

$$A_G(u,v) = \begin{cases} 1 & \text{if } \{u,v\} \in E(G) \\ 0 & \text{if } \{u,v\} \notin E(G). \end{cases}$$

If a graph is undirected then associated adjacency matrix is symmetric. For graphs without loops the diagonal entries are zeros. Adjacency matrix can also represent multigraphs if the entry $A_G(u,v)$ contains the number of edges from u to v. As far as weighted graphs are concerned, a positive element of A_G can represent the edge with weight equal $A_G(u,v)$.

Theorem 1.4.12 *Let G and H be directed graphs on the same vertex set whose adjacency matrices are A_G and A_H respectively. Then they are isomorphic ($G \simeq H$) iff there is a permutation matrix P such that $P^T A_G P = A_H$.*

Every row and column of permutation matrix contains precisely a single 1 with zeros everywhere else. There are $n!$ permutation matrices of size n. Symmetric adjacency matrix (undirected graphs) has several desirable properties. For instance it possesses real eigenvalues and eigenvectors. In addition the set of eigenvectors forms orthonormal basis. The permutation matrix P is nonsingular and orthogonal. The transformation $P^T A_X P = A_Y$ is an example of matrix similarity transformation.

Powers of adjacency matrix are related to graph walks [78] (see Theorem 1.4.13). The number of walks of given length from one vertex to another can be computed by matrix multiplication.

Theorem 1.4.13 *Let A_G be the adjacency matrix of graph G. Then $(A_G^k)(u,v)$ is the number of walks of length k starting at vertex u and ending at vertex v.*

Proof (*by induction*)
For $k = 1$ $A_G(u,v)$ is a number of edges outcoming from u and incoming to v, i.e., number of walks of length 1 starting at u and ending at v. Assume the statement is true for $k = m$.
Adding $m + 1$ gives

$$A_G^{m+1}(u,v) = (A_G^m A_G)(u,v) = \sum_r A_G^m(u,r) A_G(r,v)$$

$A_G^m(u,r) A_G(r,v) \neq 0$ if and only if $A_G^m(u,r) \neq 0$ and $A_G(r,v) \neq 0$. On the basis of assumption $A_G^m(u,r)$ is the number of walks of length m from u to r. Additionally, $A_G(r,v) \neq 0$ if and only if there exists the edge $\{r,v\}$. The non-zero value of $A_G(r,v)$ lengthens the walk of length m by 1 yielding the number of walks from u to v through r. Summing walks of length $m+1$ from u to v through all r's, we get the total number of walks of length $m + 1$.

□

Incidence matrix

Here we present the graph representation, which is a rectangular matrix [78].

Definition 1.4.14 *Incidence matrix M_G of a graph G is the $\{0, \pm 1\}$-matrix with rows and columns indexed by the vertices and edges of G, respectively, such that*

$$M_G(u,e) = \begin{cases} 1 & \text{if } e \text{ is edge outcoming from vertex } u; \\ -1 & \text{if } e \text{ is edge incoming to vertex } u; \\ 0 & \text{otherwise.} \end{cases}$$

Laplace matrices

Study of graph Laplace matrices is the main concern of spectral graph theory [78, 142]. Essentially, they are defined for undirected graphs, however, using incidence matrices it is possible introduce Laplacians for directed graphs as well.

Definition 1.4.15 *Let k_i be a degree of the i-th vertex of undirected graph $G = (V(G), E(G))$. Combinatorial Laplace matrix L_G is a matrix for which*

$$L_G(u,v) = \begin{cases} k_u & \text{if } u = v; \\ -1 & \text{if } (u,v) \in E(G) \quad (u \text{ is adjacent to } v); \\ 0 & \text{otherwise.} \end{cases}$$

The combinatorial Laplacian is related to adjacency matrix. If D_G denotes diagonal matrix of vertex degrees then $L_G = D_G - A_G$. The Laplacian is also associated with incidence matrix. The relation between incidence matrix M_G and L_G is $L_G = M_G M_G^T$. In fact, this equality can be regarded as more general (i.e. including directed graphs) definition of Laplacian [78]. The combinatorial Laplace matrix has several useful algebraic properties, for instance

- symmetry,
- singularity,

  One can observe that the sum of all Laplacian rows gives zero vector. Therefore every row (column) can be expressed as linear combination of remaining rows (columns), so the Laplacian is not a full rank matrix.
- weak diagonal dominance ($|L_G(i,i)| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |L_G(i,j)|$),
- being positive semidefinite (non-negative eigenvalues, $\forall x \in \mathbb{R}^{n \times 1} \colon x^T L_G x \geq 0$).

  Definition 1.4.16 *Let k_i be a degree of the i-th vertex of undirected graph $G = (V(G), E(G))$. With the assumption that graph G does not possess isolated vertices ($k_u \neq 0 \wedge k_v \neq 0$), normalized Laplace matrix $\mathcal{L}_G$ is a matrix for which*

$$\mathcal{L}_G(u,v) = \begin{cases} 1 & \text{if } u = v; \\ \dfrac{-1}{\sqrt{k_u k_v}} & \text{if } (u,v) \in E(G) \quad (u \text{ is adjacent to } v); \\ 0 & \text{otherwise.} \end{cases}$$

The relation between normalized Laplacian [142], combinatorial Laplacian and adjacency matrix is as follows

$$\mathcal{L}_G = D_G^{-1/2} L_G D_G^{-1/2} = D_G^{-1/2}(D_G - A_G)D_G^{-1/2} = I - D_G^{-1/2} A_G D_G^{-1/2},$$

where D_G is a diagonal matrix of vertex degrees, A_G - adjacency matrix, L_G - combinatorial Laplacian. Once again we can define normalized Laplacian as $\mathcal{L}_G = \mathcal{M}_G \mathcal{M}_G^T$, where $\mathcal{M}_G$ is normalized incidence matrix with entries scaled appropriately to vertex degrees, i.e., each entry (u, e) is multiplied by $\frac{1}{\sqrt{k_u}}$.

Normalized Laplacian $\mathcal{L}_G$ inherits the great part of properties of combinatorial Laplacian L_G. Additionally owing to normalization, all its eigenvalues are between 0 and 2 inclusive.

Part I

Background

Chapter 2

Graph analysis

Research questions regarding graphs gave rise to analytic methods that can be grouped into several categories. A brief description of core problems is presented below.

2.1 Graph structural properties

The key question how to capture quantitatively relevant structural properties of graphs provoked development of various graph measures. Throughout this work we call them graph descriptors although different nomenclature (graph invariants, graph features, graph metrics or topological measures) also exist in literature. We understand graph descriptor as scalar or vector attached to the graph as a whole or to basic graph elements, i.e., vertices or edges. The generation of graph descriptor associated with a graph is called graph embedding. The formal definition of graph descriptors in provided in Section 2.2.2. Diverse origins of graph descriptors contribute to their large quantity and several ways of grouping. See Appendix A for definitions of descriptors mentioned below as examples.

2.1.1 Descriptor target

The first distinction of graph descriptors is derived from the graph primitive they describe.

Graph descriptors describe graph as a whole. Typically they reflect a single topological property and cannot be used to reconstruct graph univocally. Examples are as follows: *density*, *diameter*, *radius*, *average path length*, *efficiency*, *average degree*, *graph clustering coefficient*, *Laplacian matrix spectrum*, *Estrada Index*, *fractal dimension* [25, 50, 63, 141].

Vertex descriptors are used to indicate vertex importance called centrality or assess other vertex-related topological feature. Frequently, they are computed by applying an iterative procedure on graph resulting in steady state and scalars assigned to its vertices. Random walks on graphs are a typical example of such an iterative procedure. The representatives of this class of descriptors are *degree*, *clustering coefficient*, *betweenness centrality*, *random walks betweenness centrality*, *Page Rank*, *communicability betweenness* [66, 117, 120].

Edge descriptors reflect importance of an edge in a certain dynamical processes on a graph or are values of two argument function on edge ends vertex descriptors. Examples include *edge connectivity*, *range of edge*, *edge betweenness* [72].

Pair descriptors are assigned to a pair of graph vertices or edges. The typical examples here include vertex-vertex similarity or dissimilarity measures such as *shortest-path length*, *longest-path length*, *f-communicability* or *commute time* [65, 123].

Last three types of descriptors can be treated in common as **element descriptors**. Given single graph, the distribution of selected element descriptor may be used to compute permutation invariant graph descriptor (see Section 2.1.5).

2.1.2 Domain descriptors

Graph descriptors have been commonly developed in the context of their application in a specific domain. The main sources of graph topological measures are listed below. The classification is imprecise as some descriptors may have been introduced in one domain and then adopted in a different one.

Classical graph theory

Some graph descriptors as *degree*, *path length* or *diameter* are considered as basic notions of classical graph theory. Their definitions were formulated in the early stages of this research field. A profound study of such descriptors produced abundant insights that have prominent applications. For instance, the distributions of vertex degrees and path lengths play significant role in theory of complex networks, having an impact on dynamical phenomena on networks. The notion of degree appears also in Euler's work on *Seven Bridges of Königsberg* in which Euler proved that a necessary condition for the existence of Eulerian cycles is that all vertices in a graph have an even degree.

Theory of complex networks

This interdisciplinary research area generated a vast amount of graph topological measures that play crucial role in exploring the effect of structure on dynamics and investigating connections between structure and functions of a network. The great part of systems described as a complex network revealed common structural and dynamical properties such as *scale-freeness*, *small worldliness*, *modularity*, *assortativity* or *disassortativity*. These properties are captured using graph descriptors like *clustering coefficient*, *average path length*, *average nearest neighbors degree*, *efficiency*, *transitivity*, *motif Z-score* [30].

Sociology

Graphs are used in sociology to encode relations between members of a population. The vertex of social network usually models an individual (or a group of individuals) and the basic concern of network analysis is to evaluate the relative importance of a vertex within the network. This has lead to the development of several vertex descriptors called centrality measures: *betweenness centrality* [71], *random walks betweenness centrality* [117], *closeness* [157] or *eigenvector centrality* [118].

Chemistry

A number of graph descriptors were derived from the topological studies of molecules. The examples are *Wiener index* [163], *normalized Wiener index*, *Platt index* [121], *Gordon-Scantleburry Index* [79]. Generally, they reflect branching of the molecue and are correlated with its Van der Waals surface.

Structural pattern recognition

Representing patterns as graphs (see Section 3.2) posed a new problem of measuring similarity between objects non-vectorial in nature. This gave rise to the development of new graph matching algorithms, including ones based on graph to feature vector transformation. Such transformation can be performed with the use of graph descriptors of any type, however, algebraic graph descriptors proved their superior robustness in many cases [164, 166].

2.1.3 Spectral descriptors

Graphs can be represented using matrices of several types (see Section 1.4). The group of well known graph matrices contains *adjacency matrix*, *incidence matrix*, *Laplace matrix*, *normalized Laplace matrix* [44, 78] and *heat kernel* [165, 166]. The study of graph matrices originating in algebraic graph theory has lead to valuable findings about connections between algebraic and structural properties of matrices and corresponding graphs. Particularly, the analysis of spectral decomposition of Laplace matrices L_G and $\mathcal{L}_G$ brought interesting results. We present a part of them in Table 2.1. The set of eigenvalues and eigenvectors contain full information about graph structure and can be used to embed graph [107, 164, 166].

This type of graph feature generation is slightly different from gathering well-known scalar descriptors, as we have plenty of available numbers (e.g. eigenvalues) but only a small part of them has established connection with a graph structure. The descriptors created using such blind procedure exhibit their advantage in the field of structural pattern recognition.

Theorem	Matrix	Structural property	Expression	Remarks
Matrix Tree Theorem	Combinatorial Laplacian L_G	Number of spanning trees nst	$nst = \frac{\lambda_1 \lambda_2 \dots \lambda_n}{n}$	n - graph size
	Combinatorial Laplacian L_G and normalized Laplacian $\mathcal{L}_G$	Number of connected components ncc	ncc = number of nonzero eigenvalues of L_G ($\mathcal{L}_G$)	
	Normalized Laplacian $\mathcal{L}_G$	Possessing bipartite component	if and only if $\sigma_n = 2$	generally $\sigma_i \leq 2$
	Combinatorial Laplacian L_G	Degree d of any vertex of G	$d + 1 \leq \lambda_{max} = \lambda_n$	
Cheeger's inequality	Combinatorial Laplacian L_G	Isoperimetric number (Cheeger constant) ϕ	$\phi_G \geq \lambda_2 \geq \frac{\phi_G^2}{2d}$	(see A.14), d is upper bound of any vertex valence

Table 2.1: Theorems joining spectrum of Laplacian with graph structure

2.1.4 Local and global descriptors

This distinction is directly linked with element descriptors. Local descriptors are computed on the basis of elements close to a given element (*k-level neighbors*). The global ones quantify property of a selected element from the whole graph perspective. The examples from each family are presented below.

Local descriptors : *degree*, *vertex clustering coefficient*, *local efficiency*

Global descriptors : *Page Rank*, *betweenness centrality*, *vertex eccentricity*

2.1.5 Statistical descriptors

The distributions of element descriptor values bring significant portion of information about graph structure. For that reason statistical moments or Shannon entropy can be used to generate valuable graph descriptors. The examples of statistical descriptors are *graph clustering coefficient*, *characteristic path length* or *average degree*. Statistical descriptors are useful for comparison of different size graphs.

2.2 Graph matching

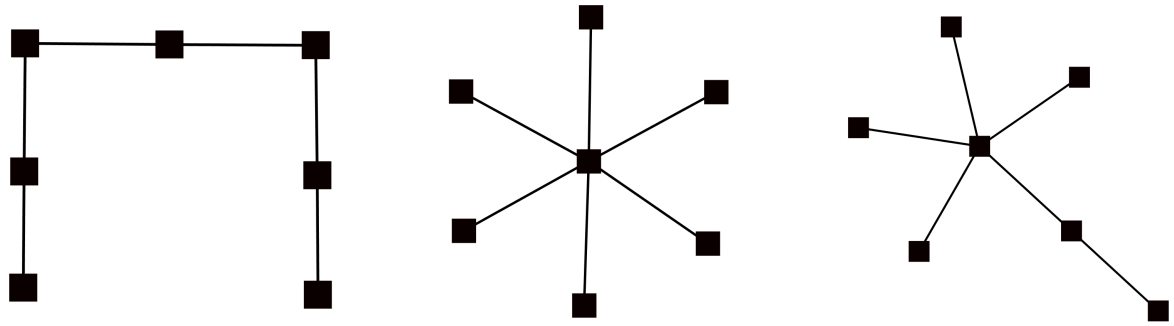
Capturing structural similarity between graphs is a task which can be approached in several ways. Before reviewing the methods of graph matching we address general issues connected with this problem.

2.2.1 General remarks

Graph is a combinational, orderless object, most frequently possessing non-trivial structure. Therefore, when considering graph matching[1], we should firstly pinpoint what we are actually going to examine. Do we focus on a single structural feature or do we plan to perform a general comparison? Each type of matching may bring specific pitfalls. Choosing single criterion can result in judgments which are far from expectations. The example of such a problem is presented in Figure 2.1. Three graphs with $n = 7$ vertices and $m = 6$ edges share the same average vertex degree. While star graph b is indeed similar to star-like graph c, the same similarity rank is obtained for path graph a, which is structurally different. Therefore, to derive valuable conclusions about affinity of two graphs one has to select the right set of comparison criteria, i.e., relevant graph descriptors. In this work we address the issue what relevance of graph features means. By selecting more graph features, we gain more general comparison but the question is where to stop and whether we need such a deep analysis. Correlations between topological descriptors make a part of them redundant, therefore to reduce computational complexity, the proper feature selection should be performed. Too general type of graph comparison is not always adequate, as in some cases we are interested in a small set of structural characteristics and exact graph matching is not needed. A typical example is confronting simulation results with real world-data, where a small set of comparison criteria is usually predefined. Summing up, graph comparison task requires balance of generality and specifity created in the context of particular application.

Representing graphs being combinatorial objects using data structures that introduce ordering of vertices, e.g., matrices, vectors or lists poses a problem of structural equivalence of graphs. Depending on selected vertices sequence, we can encode the same graph to different representations. Determining if two graphs are actually the same and finding mapping between

[1] As noticed in PhD thesis of K.M. Borgwardt [33] a subtle semantic difference exisits between notions *graph matching* and *graph comparison*. The graph comparison gives a result in the form of similarity or dissimilarity measure while graph matching focuses on finding correspondences between graph elements. Normally graph matching can be used to generate similarity/dissimilarity measures while graph comparison cannot be employed for graph matching. Nevertheless, in this work we will use these notions interchangeably, as most frequently in literature.



a b c

Figure 2.1: Three graphs with $n = 7$ vertices and $m = 6$ edges. The average vertex degree $\overline{k} = \frac{2m}{n} = \frac{12}{7} \approx 1.71$. Taking into account single comparison criteria $\overline{k}$ we find graphs a and b similar despite their significant structural diversity. Graph c, which is structurally closer to a receives the same similarity rank.

their vertices, i.e., *isomorphism* is called *exact graph matching*. *Graph isomorphism* is a crucial topic in the graph matching research field (see Section 2.2.2), however *exact graph matching* itself has limited applications. On the other hand, the related problem of *subgraph isomorphism* is not only academic but also practical.

A wide range of applications is available for *inexact graph matching* algorithms frequently referred as *error tolerant graph matching* [46]. The output of this class of methods allows for definition of graph similarity or dissimilarity measures. Let us consider some issues eccountered while performing inexact matching of two graphs: $G_1(n_1, m_1)$ and $G_2(n_2, m_2)$. In essence, approximate graph matching can be implemented in three ways.

Common subgraph

Given G_1 and G_2 we look for their maximum common subgraph and its size is returned as an output. This approach is intuitive but real-world graph patterns often suffer from structural perturbations caused by noisy data, therefore even for structurally similar patterns the size of maximum common subgraph can be surprisingly small.

Graph edit distance

Basic graph edit operations are defined: *vertex addition/deletion*, *edge addition/deletion* and the cost of each one of them is established. The algorithm determines the lowest cost sequence of operations that transform G_1 to G_2. This way of comparison is less sensitive to noise than the previous one but requires setting up of proper cost function. Moreover, as shown in Figure 2.2, graph edit distance may not take into account important structural differences.

Graph embedding

The relevant topological descriptors of both graphs are extracted. Feature vector can be used to construct similarity or dissimilarity measures. Let us take into consideration graphs depicted in Figure 2.2 and feature vector built of two graph descriptors *efficiency* and *normalized Wiener index* (see Appendix A). The pattern vectors for a, b and c are respectively $[0.267, 0.582]$, $[0.226, 0.855]$ and $[0.252, 0.580]$. In this case the structural difference between a and b was captured correctly.

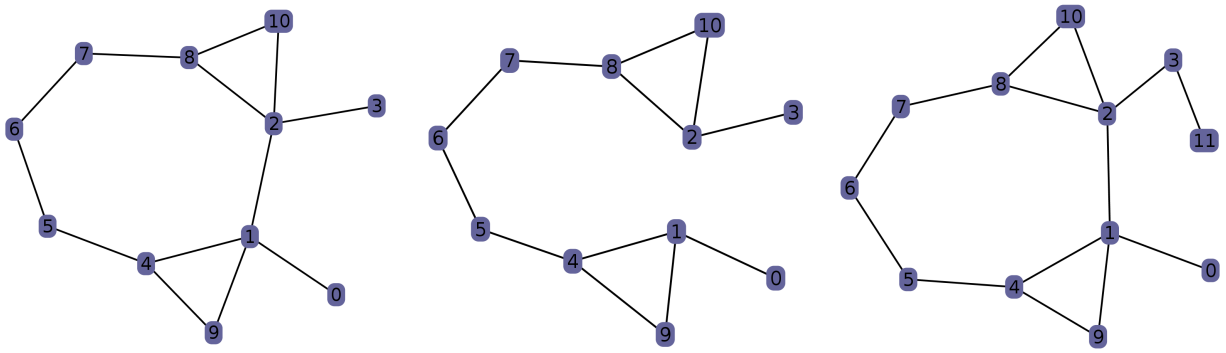


Figure 2.2: Edit operations on graphs and their cost: a. core graph, *diameter:* 4, b. deletion of single edge $(1, 2)$, *diameter:* 8, c. addition of one vertex and one edge $(3, 11)$, *diameter:* 5. Let us define cost of edge addition/deletion and cost of vertex deletion/addition as 1. The graph edit distance between a and b is 1, whereas distance between a and c equals 2, despite the fact that a is structurally closer to c.

Additionally, the relations between number of vertices and number of edges in G_1 and G_2 should be taken into account with caution. Especially the case of $n_1 \neq n_2$ or $m_1 \neq m_2$, i.e., comparing objects of different dimensionality is worth deeper consideration. The first question is to what extent the number of vertices and the number of edges should be discrimination factors. An extensive part of graph descriptors explicitly or implicitly depend on graph size and *density*, and this relation is often non-linear (see Figure 2.3). Therefore by selecting such set of features, one implicitly involves graph-size criteria that in some cases are of third-rate importance or even irrelevant. What follows, to perform purely structural comparison, one should select graph descriptors that do not depend on a graph size or to reduce the influence of dimensionality by performing normalization. If $n_1 \gg n_2$ or $m_1 \gg m_2$, graph matching is pointless, as objects are a priori so different that it is impossible to capture any similarities. The case of $n_1 \approx n_2$ and $m_1 \approx m_2$ or $n_1 = n_2$ and $m_1 = m_2$ allows for meaningful comparison.

The example of problems encountered when dealing with graphs of different density and descriptors dependent on graph size, is presented in Figure 2.3. To start with, we recall two graph descriptors *efficiency* and *Wiener index*. For the definition see Appendix A.1 and A.2. The *efficiency* is the normalized harmonic mean of graph shortest paths, while *Wiener index* is the sum of all-pair shortest-paths lengths. Let us consider four types of graphs: 3D mesh (see Figure 2.4a), balanced 3-tree (Figure 2.4b), 2D mesh (Figure 2.4c) and balanced binary tree. For each family, a set of graphs with increasing size is generated (3D meshes are available with i^3, $i \in \mathbb{N}$ vertices, 2D meshes - with i^2 vertices). In Figure 2.3 the values of *efficiency* and *normalized Wiener index* versus graph size (logarithmic axis) for four types of graphs are depicted. From these charts the following observations can be drawn.

1. The values of descriptors decay non-linearly with graph size. In fact, they tend to 0 as number of vertices grow, because of normalization factors (see the denominators in Equations A.1 and A.3) that grow faster than numerators. Therefore, both descriptors become less sensitive with increasing graph size and as a result are useless for large graphs of this type.

2. The lines on both charts intersect, hence the conclusion about similarity of graphs is different for different graphs sizes. Let us consider the values of descriptors for graph sizes

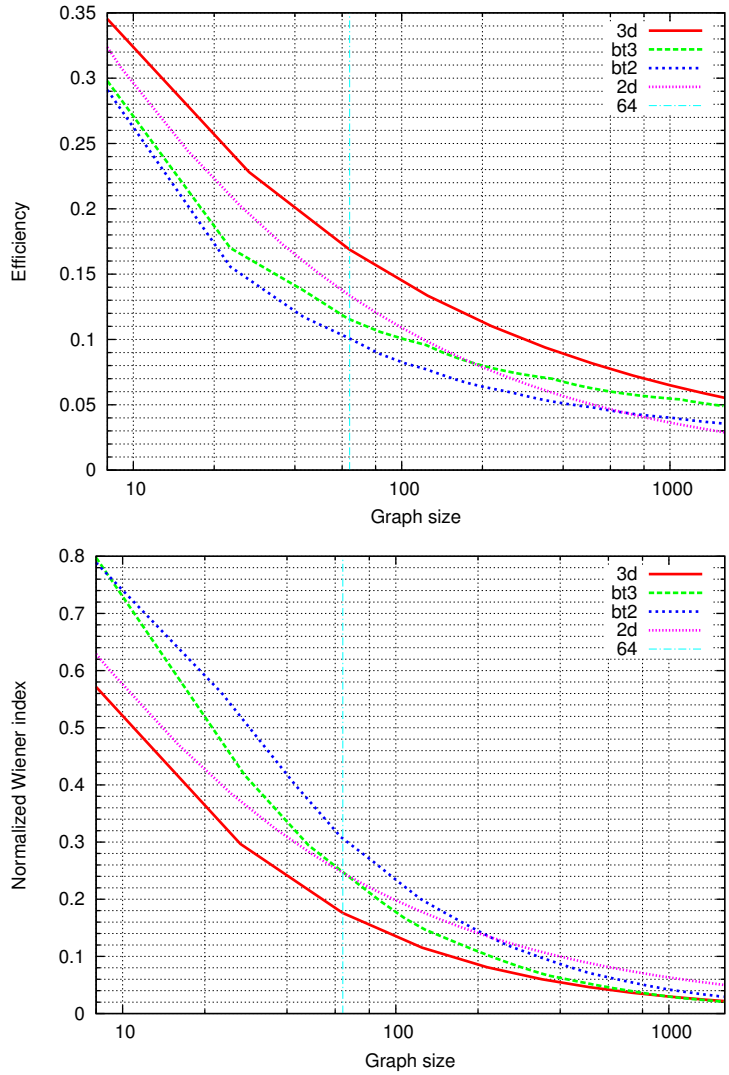
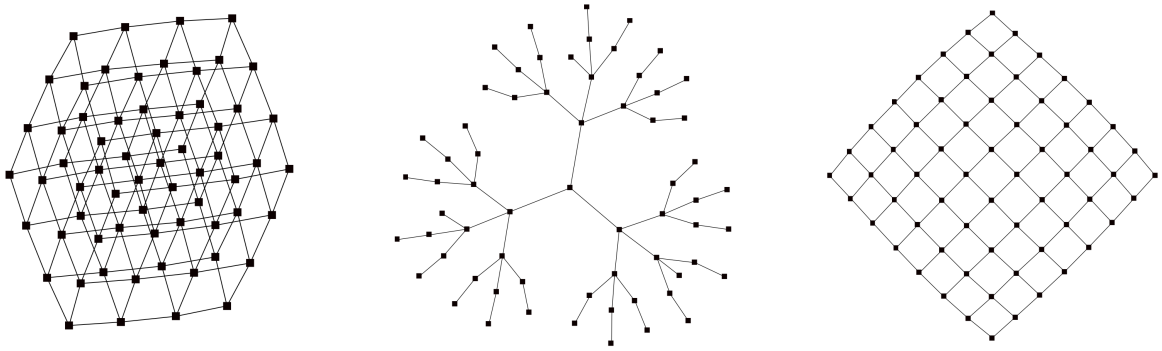


Figure 2.3: The values of *efficiency* and *normalized Wiener index* versus graphs size for graphs: `3d` - 3D mesh, `bt3` - balanced 3-tree, `bt2` - balanced binary tree, `2d` - 2D mesh. Vertical line labeled 64 was drawn to indicate graphs with 64 vertices, which are analyzed in the text.

64 and ≈1000 (see Table 2.2). The structural diversity of presented graphs is significant (compare for instance number of edges), nevertheless using two computed descriptors one cannot get coherent findings. For graph size 64, using *efficiency* we can conclude that *bt3* is closer to 2D mesh than to 3D mesh but for graph size ≈1000 the conclusion is contrary. Furthermore, from *normalized Wiener index* values we derive that for graph size 64 *bt3* is nearly same as 2D mesh while for number of vertices approximate to 1000 *bt3* is very close to 3D mesh.

The inconsistent results obtained for given descriptors reveal that proper selection of topological measures is essential for graph matching.

The methods for approximate graph matching can be also divided into pairwise methods and ones using precomputed vectorial representation (see Figure 2.7). The former approach requires two objects to perform comparison and therefore is less convenient than the latter one. Feature vectors can be used as metric-space graph representatives that enable application



a b c

Figure 2.4: Three graphs with 64 vertices: a. 3D mesh, 144 edges, b. balanced 3-tree, 63 edges, c. 2D mesh, 112 edges.

	Vertices	*Efficiency*	*Normalized Wiener index*	Edges
3d	64	0.1687	0.1758	144
2d	64	0.1336	0.2462	112
bt3	64	0.1153	0.2478	63
3d	1000	0.0649	0.0297	2700
2d	1024	0.0360	0.0624	1984
bt3	1010	0.0547	0.0295	1009

Table 2.2: The values of *efficiency* and *normalized Wiener index* for graphs: `3d` - 3D mesh, `bt3` - balanced 3-tree, `2d` - 2D mesh.

of statistical pattern recognition and data mining algorithms. Besides, the graph matching algorithms can be parametrized or not, what considerably influences their applicability. A more detailed description of these types of graph matching algorithms is provided in Section 2.2.3.

2.2.2 Isomorphism and graph canonization problem

Following definition 1.4.9 from Section 1.4 and general remarks from Section 2.2.1 we provide extended description of problems related to exact graph matching. We start with several definitions.

Definition 2.2.1 *Let $G = (V(G), E(G))$ be the graph without multiple edges. An automorphism of graph G is an isomorphism between two copies of G, i.e., it is a mapping β between pairs of vertices such that $(u, v) \in E(G) \Leftrightarrow (\beta(u), \beta(v)) \in E(G)$. A trivial authomorphism of a graph G is an authomorphism where $\beta(v) = v$ for all $v \in V(E)$.*

Definition 2.2.2 *Let $G = (V(G), E(G))$ be the graph without multiple edges. An automorphism partition of graph G is a sequence of disjoint subsets $V_1, \ldots, V_k$ such that for all pairs of vertices u,v there exist an automorphism β where $\beta(u) = v$. The automorphism partition divides V into sets consisting of vertices that can be mapped onto one another.*

As far as the complexity class of graph isomorphism problem is concerned, it is not recognized to be in **P** or **NP-com** [69]. This result is particularly interesting, because efficient

implementations are known for many wide graph classes such as *planar graphs*, *bounded degree graphs*, *circular arc graphs* and in case of *random graphs* the problem appears to be almost always easy. However, despite of large effort put into finding polynomial general solutions of graph isomorphism, it still remains an open problem. In order to overcome this barrier and provoke different-perspective research, the new complexity class **GI-com** (graph isomorphism complete) was defined [102]. The problems such as counting number of isomorphisms between two graphs, counting the number of non-trivial automorphisms of a graph (see Definition 2.2.1) and determining the automorphism partition of a graph (see Definition 2.2.2) fall within this class.

The invariance under isomorphism and graph canonization are crucial notions that appear when considering exact matching algorithms.

Definition 2.2.3 *Let f be a function defined on graphs and let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be the graphs. f is called graph invariant iff $G \simeq H \Rightarrow f(G) = f(H)$.*

Definition 2.2.4 *Let f be a function defined on graph vertex set and let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be graphs so that $G \overset{\alpha}{\simeq} H$. f is called vertex invariant iff $\alpha(u) = v \Rightarrow f(u) = f(v)$.*

Definition 2.2.5 *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be the graphs. The graph invariant f is complete iff $f(G) = f(H) \Rightarrow G \simeq H$. Complete graph invariant is an injective mapping.*

Definition 2.2.6 *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be the graphs. The vertex invariant f is complete iff $f(u) = f(v) \Rightarrow G \overset{\alpha}{\simeq} H \wedge \alpha(u) = v$ where $u \in V(G), v \in V(H)$.*

Computing any complete graph invariant is equivalent to determining graph isomorphism. The graph invariant can have diverse form: set, multiset, matrix, vector, scalar, polynomial or string. The complete graph invariant represented by a string is usually called *canonical label* [19]. Graph canonization is a problem of particular interest of organic chemistry as it can provide unique names for complex molecules [47]. For the needs of this work we restrict the value set of a graph invariant and introduce the definitions of *graph descriptor* and *vertex descriptor*.

Definition 2.2.7 *Let f be a graph invariant such that $f : \mathbb{G} \to \mathbb{R}^n$, where $n \geq 1$ and $\mathbb{G}$ is a set of all graphs. The f is called n-element graph descriptor or alternatively explicit graph embedding.*

Definition 2.2.8 *Let f be a vertex invariant of graph $G = (V(G), E(G))$ such that $f : V(G) \to \mathbb{R}^n$, where $n \geq 1$. The f is called n-element vertex descriptor.*

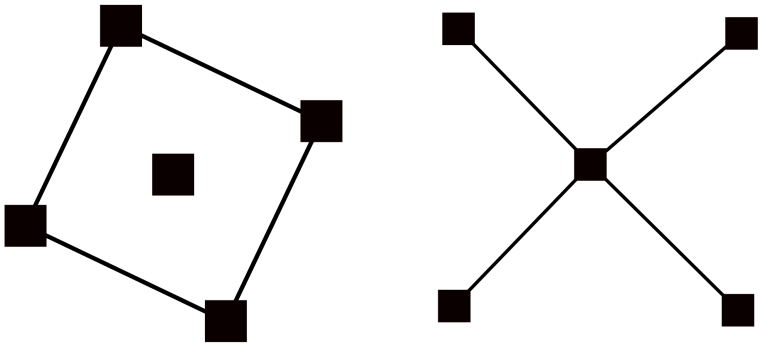
As graph isomorphism is one-to-one vertex mapping, vertex descriptors are practical tool for solving this problem. The complete vertex descriptor allows for direct construction of isomorphism by linking vertices with the same value of descriptor and therefore its computation is equivalent to solving graph isomorphism problem. Not-complete vertex descriptor can be also useful in determining graph isomorphism. In fact even simple vertex descriptors such as vertex degree are practical for wide range of graphs.

Graph spectrum and isomorphism

In this section we take into account the correspondence between graph spectrum and graph isomorphism.

Lemma 2.2.9 *Let $\Delta_A(G)$ and $\Delta_A(H)$ denote the set of eigenvalues (with their multiplicities) of adjacency matrix of graph G and H, respectively. If graphs G and H are isomorphic, then $\Delta_A(G) = \Delta_A(H)$. The analogous theorem holds for eigenvalues of Laplace matrices.*

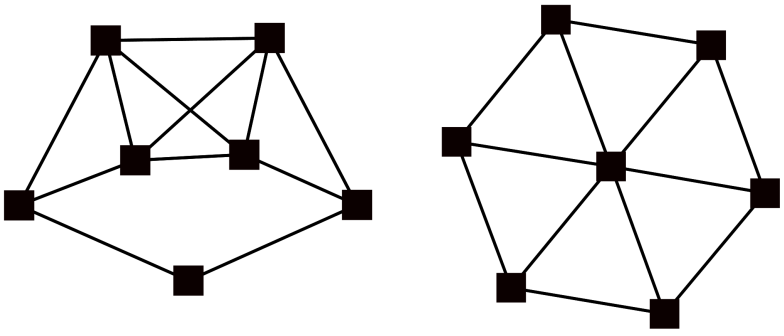
Therefore given two distinct sets of graph spectra, we can easily conclude that associated graphs are not isomorphic. Unfortunately, the inverse reasoning is not always true, i.e., there exist non-isomorphic graphs with the same spectra. Graph spectrum is not a complete graph invariant. Graphs with the same spectrum are called *cospectral* or *isospectral*. The examples of cospectral (regarding adjacency matrix), non-isomorphic graphs are depicted in Figure 2.5 (graphs G_1, G_2) and in Figure 2.6 (graphs H_1, H_2).



G_1 G_2

Figure 2.5: The example of two non-isomorphic cospectral graphs: G_1 and G_2.

For graphs G_1 and G_2 from Figure 2.6: $\Delta_A(G_1) = \Delta_A(G_2) = \{2, 0^{(3)}, -2\}$ but, interestingly as far as the spectrum of Laplacian is considered $\Delta_L(G_1) = \{4, 2^{(2)}, 0^{(2)}\} \neq \Delta_L(G_2) = \{5, 1^{(3)}, 0\}$. Therefore the cospectrality with regard to one matrix does not imply the general cospectrality.



H_1 H_2

Figure 2.6: The example of two non-isomorphic cospectral graphs: H_1 and H_2.

For graphs H_1, H_2 from Figure 2.6: $\Delta_A(H_1) = \Delta_A(H_2) = \{3.646, 1^{(2)}, -1^{(2)}, -1.646, -2\}$ and $\Delta_L(H_1) = \{4+\sqrt{3}, 5^{(2)}, 3+\sqrt{2}, 4-\sqrt{3}, 3-\sqrt{2}, 0\} \neq \Delta_L(H_2) = \{7, 5, 4^{(2)}, 2^{(2)}, 0\}$.

If one of isospectral graphs possesses a property the second does not, then it cannot be determined by the spectrum of given matrix. For instance, H_2 is planar, whereas H_1 is not, hence planarity is not spectrally-determined. Similarly G_2 is connected while G_1 is not.

Lemma 2.2.10 *Let $\lambda_1 \leq \ldots \leq \lambda_n$ be the eigenvalues of adjacency matrix A_G of graph G. Assume that λ_i is isolated, that is $\lambda_{i-1} < \lambda_i < \lambda_{i+1}$ and let v_i be the corresponding eigenvector. Let H be a graph isomorphic to G, and let w_i be the i-th eigenvector of H. Then, there exists a permutation π such that $v_i(j) = w_i(\pi(j))$.*

The eigenvectors of adjacency matrix associated to isolated eigenvalues (multiplicity 1) can be used to create vertex descriptors and then to construct isomorphism, however the success of this procedure is not guaranteed. Especially in case of strongly-regular graphs[1], the lack of isolated eigenvalues makes this approach not appropriate.

Graph isomorphism algorithms

Solution of a graph isomorphism can be expressed in two ways either as a permutation matrix or by canonical label. The first approach is pairwise while the second one resembles construction of complete graph invariant. The label of a graph can be constructed by packing columns (or rows) of adjacency matrix into one vector. After determining all automorphisms of the graph (reflected by different adjacency matrices), the vectors can be put in lexicographic order. The smallest binary vector can serve as canonical label of the graph, however finding all non-trivial automorphisms is as difficult as determining isomorphism.

Graph and vertex invariants are practical for finding structural correspondences, therefore they are common heuristic used in graph isomorphism algorithms to prune search space by rejecting pairs of graphs possessing different invariant sets or pairs of vertices whose invariants do not match [54]. The pseudo-code of the algorithm taking advantage of vertex descriptors heuristics is depicted below (Algorithm 1 and Algorithm 2). The backtracking procedure is present in commonly used algorithm for graph and subgraph isomorphism proposed by Ullmann [152]. More recently, Cordella et al. introduced space-efficient, recursive VF2 algorithm capable of dealing with large graphs [49]. In this algorithm, finding vertex mapping function is described by means of State Space Representation. On the basis of partial mapping solution s_i, the possible pairs of new, s_i-derived vertex mappings are generated and evaluated using five predefined feasibility rules [49]. Those rules take into account both structural and sematic information (attributes). If the feasibility function returns true, the new state s_{i+1} is generated, followed by recursive function call. This yields depth-first state space search with pruning provided by feasibility rules. Permutation matrix reconstruction based on Page Rank [120] vertex descriptors was presented in [80]. This polynomial algorithm allows for finding isomorphism for a class of Markovian spectrally distinguishable graphs and was shown to be more efficient than VF algorithm. In one of our works [55] we combined Page Rank vertex descriptors with spectral descriptors (see Section 2.2.2) derived from two variants of adjacency matrix. This allowed us to improve isomorphism search accuracy for difficult classes of sparse graphs. In the next work [54], we investigated how graph dynamical system can be used for generation of

[1]Graph G is strongly-regular if there exist three positive integers a, b, c, such that every vertex has a neighbors, every adjacent pair of vertices has b common neighbors, and every nonadjacent pair has c common neighbors.

vertex descriptors practical in determining graph isomorphism. Several cellular-automata-like rules are employed to update state of a given vertex on the basis of states of its neighbors. With k subsequent updates we obtain more correlated vertex states which, depending on the selected rule, may converge to the steady state. Intermediate states can be used to support isomorphism search. The algorithm perfoms well on benchmark dataset at the same time being less computationally expensive than $\mathcal{O}(n^3)$ algorithms such as Page Rank [54].

The Nauty program by Brendan McKay [1] forms a robust framework for graph canonical labelling and computing automorphism groups of graphs [113]. Vertex invariants, specified by a user can be employed to reduce size of search tree, nevertheless the selection of proper set of invariants is not straightforward as their usability varies with the type of graph.

Algorithm 1

Input: graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, n - number of vertices
Output: P permutation matrix or *NONE* if $G_1 \not\simeq G_2$

1: **for** $i = 0$ to $n - 1$ **do**
2: $invariant_1(i)$ =vertex invariant of i-th vertex of G_1
3: $invariant_2(i)$ =vertex invariant of i-th vertex of G_2
4: **end for**
5: **if** $sort(invariant_1) \neq sort(invariant_2)$ **then**
6: **return** *NONE*
7: **end if**
8: reorder vertices of G_1 and $invariant_1$
9: $S = \{\}$
10: **if** *isomorph(S,1,P)* **then**
11: **return** P
12: **else**
13: **return** *NONE*
14: **end if**

Algorithm 2 *isomorph*

Input: set S, integer k, permutation matrix P
Output: **true** if isomorphism can be constructed for entire input graphs

1: **if** $k = n + 1$ **then**
2: **return true**
3: **end if**
4: **for all** $j \in V_2 \backslash S$ **do**
5: **if** $(invariant_1(k) \neq invariant_2(j)) \vee \neg$*can_match(k,j,P)* **then**
6: **continue**
7: **end if**
8: $P(k, j) := 1$
9: **if** *isomorph*$(k + 1, S \cup \{j\})$ **then**
10: **return true**
11: **end if**
12: **end for**
13: **return false**

2.2.3 Overview of graph comparison algorithms

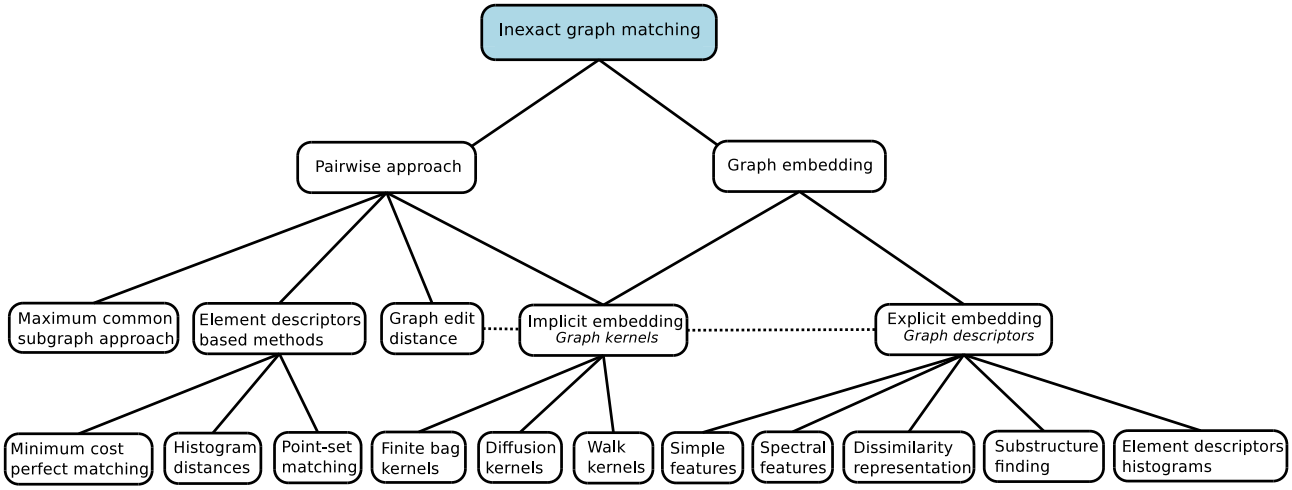


Figure 2.7: Types of inexact graph matching algorithms

Inexact graph matching can be approached in various ways. Schematic view of the most frequently used techniques is depicted in Figure 2.7. Owing to their flexibility, the algorithms based on the notion of graph edit distance play prominent role among classical pairwise methods. Unfortunately, their exponential computational cost is prohibitively high for medium-size and large graphs [138]. Conveniently, sub-optimal algorithm based on bipartite graph matching, running with time complexity $\mathcal{O}(n^3)$ was presented in [133]. Graph edit distance is related to the size of maximum common subgraph via specific cost function [39] and this association can be employed to construct robust graph metric [40].

The group of graph comparison methods that have recently gained considerable attention rely on explicit or implicit embedding of graphs into metric space. The former approach assigns feature vector to a graph yielding linear space with Euclidean distance. The latter one builds graph kernel that defines metric in high-dimensional feature space based on graph primitives such as walks, paths, cycles or subtrees [73]. The focus of this work is explicit graph embedding therefore in the next paragraph an overview of related literature is presented.

Vectors representing graphs should be invariant under graph isomorphism. This crucial prerequisite allows for moving from orderless graph space to ordered algebraic domain. Typical graph representations such as adjacency matrix, Laplace matrix or neighborhood list depend on vertex ordering, therefore they are not identical for the same graphs with permuted labels. Several methods have been used to provide invariance of graph embeddings under isomorphism. In the most straightforward approach, fixed-order enumeration of scalar descriptors such as *efficiency*, *diameter* or *clustering coefficient* [50], forms a vector representation [52]. Simple collecting of graph characteristics including degree distribution measures or node centrality correlations is present in biological networks comparison, where topological measures are computed to understand functions of structural elements [63, 169]. When unique vertex labels are known *a priori* the construction of vector representation is easier as permutation invariant functions are no longer necessary. In this case graph feature vector can be obtained from sequence of vertex or edge descriptors. This is particularly useful for comparison of metabolic networks whose nodes represent chemical compounds. For instance, construction of phylogenetic tree based on network embeddings was presented in [17, 18]. A more general approach for

extracting graph characteristics uses permutation invariant functions, e.g. symmetrical polynomials [164]. Also aggregated statistics of graph-element descriptors such as mean values and standard deviations of vertex degrees, edge betweenness centralities, shortest paths lengths or commute time metrics [123], are employed to generate meaningful graph features [53, 56]. The information about frequencies of element features can be directly included into graph pattern vector by aggregating histogram bins. Such an approach is presented in [32], where different binnings are tested and histograms are additionally normalized by the graph volume to get representation independent of graph size. In the next method, the features are extracted on the basis of distances to prototype graphs giving dissimilarity graph representation [134]. Selection of right distance measure and good prototypes is a main issue that should be addressed here.

Capturing relevant and discriminative structural properties of graphs that in addition would be robust to structural noise is a challenging task. The scalar descriptors or simple statistics are most frequently not strong enough to analyze more difficult datasets, like for instance IAM benchmark database [132]. Elegant methods of graph features generation use invariants computed on the basis of spectral decomposition of graph matrices. Eigenvectors and eigenvalues of Laplace matrix, forming truncated modal matrix, were used in [107] to extract per-eigenmode characteristics like eigenmode volume or eigenmode perimeter. High-dimensional pattern vectors, obtained from spectral matrix using symmetrical polynomials, which provide invariance under graph isomorphism were introduced in [164]. These representations are rich enough to distinguish Delaunay graphs representing rotating 3D objects. Heat kernel matrix, obtained by exponentiating the Laplacian eigensystem encodes time-scaled information about diffusion process on a graph, therefore it forms productive basis for graph embedding algorithms. For instance, Xiao and Hancock construct robust graph characteristics by computing permutation invariants on heat kernel trace [166]. The pattern vectors obtained in this way can be tuned by time parameter, which allows for navigation between local and global features. A different approach is presented by the same authors in [165], where vertices of a graph are embedded into vector space using Young-Householder decomposition of heat kernel matrix. Next, Mahalonobis distance between obtained node positions serves as graph dissimilarity measure. In the work by Jouilly and Tabbone [93], dissimilarity representation, computed on the basis of vertex signatures, is transformed to a set of feature vectors with the use of constant shift embedding technique. Such an approach allows for moving from pairwise similarities to vector space. More recently, polynomial coefficients from Ihara zeta function were proposed as a source of information about cycle structure of a graph [131]. They were used to build low-dimensional, expressive feature vector outperforming older spectral descriptors in unweighted and weighted graph recognition.

Graph kernels form general framework for similarity-based structured data analysis. Provided that graph kernel k is definite positive (d.p.), we obtain Hilbert space embedding with the scalar product defined by k. With this approach linear classifiers in Hilbert space $\mathcal{H}$ become non-linear in original pattern space. The most straighforward way to construct d.p. graph kernels is by using graph descriptors and predefined $\mathbb{R}^n$ kernels such as linear, polynomial or Gaussian kernel. The significiant part of graph kernels is based on R-Convolution approach, in which two graphs are decomposed into substructures and similarity functions computed for all pairs of these substructures are employed to obtain final graph kernel [153]. Counting number of matching walks in two labeled graphs with a help of *product graphs* allows for building product graph kernel, which can be computed as a matrix exponential/power series with time complexity $\mathcal{O}(n^6)$ [73]. The related marginalized graph kernel takes into account all pairs of labeled random walks from two graphs and computes expectation of nonnegative walk kernel

defined using simple vertex and edge kernels [96]. These two types of random walk kernels have several drawbacks including high computational complexity $\mathcal{O}(n^6)$ and similarity score biased by short walks and nodes/edges repeating in cycles. The other R-Convolution-type graph kernels use bags of patterns such as subtrees, cycles, shortest-paths or limited depth-first search paths [33]. In the diffusion kernels approach, graph edit distance or other similarity/dissimilarity measure is employed to construct similarity matrix which is then transformed into d.p. form using appropriate decay factors and infinite matrix series such as *exponential diffusion kernel* [116].

Chapter 3

Graph data

In this chapter we present sample graph data from different research fields and explain practical problems they pose. We provide more detailed description of those networks which will be used in experiments described in the second part of this dissertation. Nevertheless, before coming to overview of structured data applications in contemporary science we introduce short note about types of graphs.

3.1 Types of graphs

Graph vs. network

In this work we assume that these two terms have the same meaning. However, the notion *network* will be used when we aim to put emphasis on the fact that it refers to some real-world data that evolve with time. We refer to object as *graph* whenever it is treated more like mathematical concept or data structure.

Directed and undirected graphs

Undirected graphs model symmetric relations whereas digraphs asymmetric ones. The sample of digraph is a gradient network in which each node has a potential and a directed edge points towards neighbor with a smallest potential. Gradient networks are used to study jamming mechanisms in transportation systems [149].

Weighted graphs

Weighted graphs allow for encoding heterogeneous relations. The edge weight can be normalized, so that its value ranges from 0 to 1. Typical example of weighted network is correlation graph such as brain functional organization network [144] or gene co-expression network [168].

Labeled graphs

Vertex or edge labels provide additional semantic information which can be involved in matching process. Presence of vertex labels facilitates finding vertex correspondences and in case they are unique, determining graph isomorphism is a trivial task. Typical example of labeled real-world graph is metabolic network with vertices denoting chemical compounds (see Section 3.3.1).

Random graphs

We understand *random graph* with n vertices and m edges as a graph created by con-

necting pairs of randomly selected nodes until actual number of edges reaches m. In alternative definition the probability $0 < p < 1$ of connecting each pair of vertices is used as a model parameter. These types of graphs were extensively studied by Paul Erdős using probabilistic methods and currently are one of the best explored classes of graphs. For large values of n the degree distribution of *random graph* is Poisson with mean value $\langle k \rangle$

$$P(k) = \exp(-\langle k \rangle) \frac{\langle k \rangle^k}{k!}. \tag{3.1}$$

Let $P(k'|k)$ be a probability that a vertex of degree k is connected to a vertex of degree k' and $k_{nn}(k)$ is average degree of nearest neighbors of vertices with of degree k. Erdős *random graphs* are uncorrelated, that is $k_{nn}(k)$ an $P(k'|k)$ do not depend on k [30]. The *clustering coefficient* (see A.3) of this type of graphs tends to 0 with growing n and in general *random graphs* are poorly clustered. The described models were expected to reflect real-world structures, however soon it was found that networks constructed from experimental data are most frequently different from random ones, possessing non-Poisson degree distributions and correlated vertex degrees.

Scale-free networks

As it was demonstrated in recent decade, real-world networks commonly exhibit inhomogeneous structure reflected by power-law degree distributions [15, 30].

$$P(k) = Ak^{-\gamma}, \tag{3.2}$$

where exponent (scaling parameter) γ is typically between 2 and 3, k is vertex degree and A is a normalization constant. For power-law $P(k)$ average values and standard deviations are not a good representatives of underlying data. The network following *scale-free* degree distribution has heterogenic topology with a few densely connected *hub* nodes and many low-degree nodes. As a consequence such a network is resistant to random attacks and vulnerable to targeted attacks, in which *hubs* are eliminated with a greatest probability. In the model of evolving network, proposed as a generating mechanism for *scale-free* networks, newly-created vertices connect to existing vertices with a probability proportional to their degree. This is rule is known as *preferential attachment*.

As the *scale-free* structure of a network potentially reveals underlying dynamics, the significant research efforts was put on finding power-laws in real-world data. Vast amount of networks was claimed to be *scale-free* including cellular networks, WWW and social networks. Nevertheless, as pointed out in the work by Clauset et al. [45], the right detection of power-laws in empirical data is complicated by considerable fluctuations appearing at the tail of distribution. Therefore, in order to get accurate results, the authors recommend using more elaborate, maximal-likelihood-based framework instead of simple least-squares log-log plot fitting. After statistically rigorous evaluation of datasets previously classified as obeying power laws, we can find a part of them as not following power-laws.

Small-world networks

The experiments on chains of correspondence conducted by Stanley Milgram initiated long-standing debate about shortest path structure of social-networks and gave rise to *six degrees of separation* idea [15]. Since then, many real-world networks was shown to possess *small-world* property, that is small average shortest path (see A.5) scaling

logarithmically with a graph size. As many types of graphs share this property (including random graphs), the more strict definition of *small-world* networks requires additionally high *clustering coefficients* (see A.3). The *small-world* property influences dynamics on a graph as it provides fast transport through short links, what is a crucial factor for such processes on networks as rumor propagation or disease spread.

3.2 Graphs representing patterns

Graph-based representations and graph learning are the core of structural pattern recognition field [46, 111]. In this section we review most common image to graph transformations and explain why we can benefit from such preprocessing. Next, a short overview of graph data in machine learning is presented.

3.2.1 Images, shapes and scene organization

Graphs allow for capturing structure of image or shape elements in a manner which is invariant under rotation, scaling, translation and changes in viewpoint. Encoding relations between scene primitives is particularly benefitial in content-based image retrieval and analysis. Therefore graph-based representations are gaining considerable attention in computer vision and image processing communities [2]. Transformation of pixel matrix into graph requires selection of objects mapped to node primitives and binary relations between them to form edges. This can be achieved in several ways.

1. Using skeleton junctions as vertices and paths between junctions as edges (binary image representation [53])
2. Treating image segments as nodes and connecting neighboring segments [103] (regional adjacency graphs)
3. Extracting corners from image, performing Voronoi tessellation and joining adjacent Voronoi cells to obtain Delaunay graph representing image [107]
4. Using Gestalt relationships between lines extracted from an image. Graph vertices stand for lines while measures of inter-line proximity, continuity and paralelism are employed to obtain weight between two line sents. This yields weighted relational image representation [34]
5. Representing 2D shapes as shock trees, which are constructed on the basis of the differential structure of the shape boundary [139]
6. Transformation of 3D shapes into graph treating critical points of differentiable function defined on surface as vertices and edges reflecting connectivity of level sets (Reeb graphs [29])

Graph representation abstracts image contents efficiently, however the number of vertices and edges vary due to noise and segmentation errors. This is why error-tolerant, inexact graph matching should be used here.

Multiple graph representation based on skeletons

In this section we present the method for obtaining multiple-graph representation of an image which was used by author in unsupervised graph learning experiments reported in work [53]. Our aim is to get skeletons on the basis of several binary counterparts of greyscale image and then to convert these skeletons to graphs. The skeleton of the shape depicted on binary image has a form of a graph drawn on paper. Such a graph is planar and not necessarily connected. The endpoints and junctions on the skeleton can be treated as graph vertices while the curves between such points as edges.

The structure of the skeleton stores an important part of information about original region such as connectivity, extent or symmetry. Therefore, it seems that converting greyscale image to its binary counterpart and skeletonization is a natural method of image to graph transformation. Unfortunately, by thresholding we loose significant amount of information contained in greyscale image. To minimize the loss we apply multiple thresholds and get multiple skeletons. The thresholds are selected arbitrarily, e.g., by choosing several values from middle-intensity range or computed via iterative method with various initial thresholds. The binary counterparts of greyscale images representing the same object obtained by thresholding at given level are similar provided that histograms of images are equalized to use full intensity range.

The next problem we face is that skeletonization is very sensitive to small changes in the binary image, especially caused by noise. In addition, by thresholding images with lots of details such as photos of real-world objects we obtain binary counterparts with ragged edges and many separate small groups of pixels. Then, as a result of skeletonization we get graph of a large size, with structure reflecting minor features of image and attenuating major ones. Thus, to amplify low-frequency components of image and reduce noise we take advantage of Gaussian blur filter, which makes the intensity surface more smooth. As a result of thresholding of such a surface we obtain less ragged shapes and, what follows, a skeleton with a smaller number of junctions. We have decided to apply Gaussian blur instead of other noise-reducing filters such as mean or median filter because it is better in preserving edges and has a separable kernel.

Let I be a greyscale image and let $T = [t_1, \ldots, t_k]$ be an array of thresholds $t_i \in (0; 255)$. The process of converting image to ordered set of graphs is performed as described on Algorithm 1.

Algorithm 3 Converting image to ordered set of graphs

Input: I, $T = [t_1, \ldots, t_k]$, σ
Output: $S = [g_1, \ldots, g_k]$ {set of graphs}

1: $I = equalizeHistogram(I)$;
2: $I = gaussianBlur(\sigma)$;
3: $i = 1$;
4: **repeat**
5: $\quad BI = getBinary(I, t_i)$;
6: $\quad SI = skeletonize(BI)$; {skeletonization by thinning}
7: $\quad g = convertSkeletonToGraph(SI)$;
8: $\quad S[i] = extractGreatestConnectedComponent(g)$;
9: $\quad i = i + 1$;
10: **until** $i > k$;
11: **return** S;

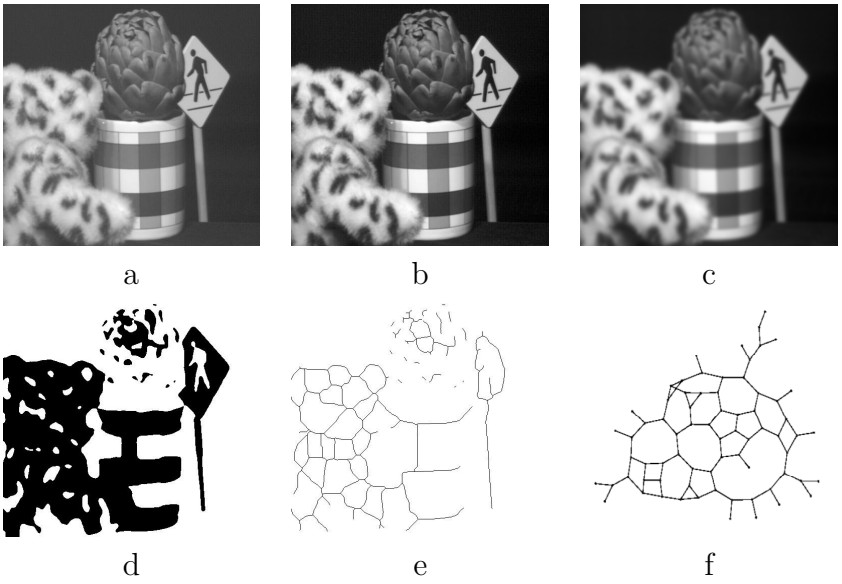


Figure 3.1: The process of image to graph transformation using skeletonization: a. original image, b. histogram equalization, c. Gaussian blur, d. thresholding, e. skeletonization, f. greatest connected component of skeletal graph (71 vertices and 90 edges).

Histogram equalization is performed by calculating cumulative frequencies within image. The parameter of Gaussian blur filter σ sets kernel size, therefore it affects the degree of smoothing and, consequently, the size of graph obtained from skeleton. Skeletonization is performed with the use of thinning morphological operation. The skeleton is converted to graph via algorithm which distinguishes between pixels with one, two and three or more neighbors treating two-neighbor pixels as edge points and remaining two kinds as vertex points [143]. The graph obtained from skeleton is usually not connected. Because we are going to use matching method that requires connected graph, we select the greatest connected component of skeletal graph as the representative for a given threshold. The drawback of the described approach is that some of information contained in the skeleton is lost. On the other side, small-sized connected components being result of noise or tiny separate segments are omitted. In Figure 3.1 the process of image to graph transformation for single threshold 124 and $\sigma = 3.0$ is presented. The greatest connected component extracted from skeleton has 71 vertices and 90 edges. As we can see this part of skeleton represents only a fragment of the photo (teddy bear and a mug). Applying different thresholds can give different results.

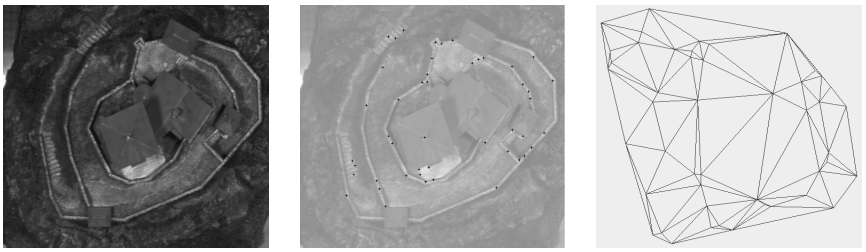
Delaunay graphs

Extracting feature points using corner detector and applying Delaunay triangulation yields purely structural image representation. This transformation is frequently used in image recognition tasks owing to its good correspondence with a structure of underlying image [107]. The schematic view of corners-based graph generation is depicted in Figure 3.2. Typically, Harris corner detector [88] is employed to obtain feature points. A corner reflects image point where intensities change rapidly in two perpendicular directions (crossing edges).

In order to add connectivity information to discrete feature set, 2D Voronoi cells are generated around each corner. Next, dual graph is constructed, in which edges join corners with adjacent Voronoi cells. Delaunay graphs possess several properties that make them well-suited

for encoding image structure [110].

- No point appears inside circumcircle of a triangle, therefore any additional noise point affects only local structure, in the region indicated by neighboring circumcircles.
- Partial occlusions leave remaining part of a graph similar to original one.
- The nearest neighbor graph for a point set is a subgraph of the Delanuay triangulation.
- Delaunay triangulation maximizes minimum angle of all triangle angles, what allows to avoid skinny triangles.
- 2D Delaunay triangulation can be computed with time complexity $\mathcal{O}(n \log(n))$, where n is number of points (divide and conquer algorithm [82]).
- Average vertex degree of Delaunay graph on plane is 6.
- Vertex degrees reflect corners density - regions with many corners yield degrees above the average.
- Delaunay graph is planar.



a b c

Figure 3.2: Image to graph transformation using Delaunay triangulation of corners: a. original image, b. applying Harris corner detector [88], c. Delaunay triangulation.

Delaunay graph derived from images can be enriched by adding node labels such as corner positions (2D attributes) or edge weights. In one of previous works [56], we studied application of weighted Delaunay graphs encoding extra information about triangles for aerial and satelite photos recognition. In order to establish edge weight we take into consideration two triangle angles facing this edge (see Figure 3.3, angles α and β). The important property of Delaunay triangulation is that $\alpha + \beta \leqslant \pi$. We define edge weight as a permutation invariant function of two angles $f(\alpha, \beta)$. For edges lying on the convex hull of point set, the second angle is 0 (unbounded Voronoi cell, second triangle vertex located infinitely far away). The weights are normalized to be in the range 0.0 to 1.0. Function f can be one of the following: $\max(\alpha, \beta)$, $\min(\alpha, \beta)$, $\alpha * \beta$, $\alpha + \beta$. For instance, by selecting $f = \max(\alpha, \beta)$ we put greater importance to edges joining vertices which posess spatially close common neighbor. In this manner additional spatial information is incorporated into graph representation. Such weights are particularly useful in aerial photo recognition as in this type of patterns relative locations of objects change rarely and angles remain approximately constant. The frequencies of angles in Delaunay graphs were also used as a feature vector representing shapes [146].

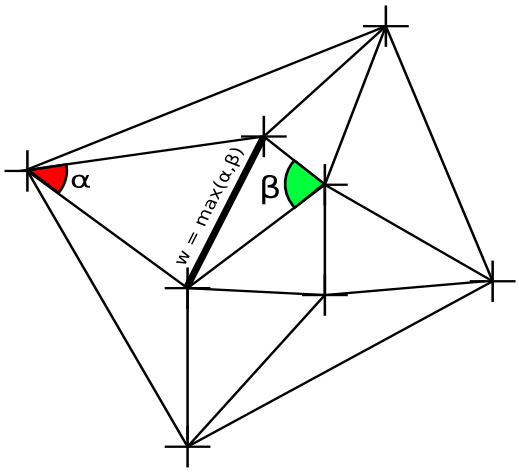


Figure 3.3: Computation of edge weights in Delaunay graph based on edge-facing angles.

3.2.2 Structural patterns and applications

In this section we present short overview of datasets, which benefit from structural pattern representation and recognition. The IAM benchmark database [132] contains graphs covering wide range of applications including recognition of digits, letters, symbol drawings and fingerprints. Apart from image representations it also stores structured web documents from different categories, chemical compounds with activity or inactivity against HIV, mutagen and nonmutangen molecues and proteins grouped according to their enzyme class. Building efficient classifiers for those data has strong practical implications. Columbia object image database (COIL) [115], which stores different views of rotating 3D objects (each 5°) is frequently used to test Delaunay graph representations in unsupervised learning tasks. Also, Caltech 256 photo set [81], which consists of objects on different backgrounds was analyzed in a strucutral manner [166].

3.3 Graphs in biology

Biological systems can be described at the cellular level as a large network integrating expression data for genes, proteins and metabolites. Interaction patterns among basic components of living cell are used to form various types of graph data, that allows for investigation of such phenomena as adaptation or robustness [24], and constitutes general framework for system biology.

3.3.1 Metabolic networks

This type of cellular network consists of metabolic reactions taking place inside living organisms to provide energy or build cell elements. The reactions are normally catalyzed by proteins called enzymes. The enzymes are synthesized using information contained in genes. The common approach in biology for the study of metabolism is the discovery of so-called metabolic pathways, which are the series of reactions that generate or utilize specific substance e.g. *glycolysis* pathway converts *glucose* into *puryvate* releasing free energy. The rate of production of molecules through metabolic pathway is called *metabolic flux*. The *flux* is regulated by enzymes in response to changing environmental conditions. By linking pathways one can form metabolic network, providing system view of cell metabolism.

The basic unit of metabolic pathway is a reaction catalyzed by enzyme in which substrates are transformed to products. As the reaction can be multi-substrate and multi-product its representation as a graph is not straightforward. In building metabolic networks from basic reactions two approaches are used. In the first one metabolites are modeled as labeled graph vertices, whereas reactions form edges. The enzymes are not directly included. This way of representation requires identification of *principal link*, i.e., a pair of substrate and product so that majority of carbon atoms in substrate is also present in product [97]. The different perspective method for integration of pathways into one metabolic network is treating enzymes as nodes of network and linking two nodes a and b if product of reaction catalyzed by enzyme a is a substrate of reaction catalyzed by enzyme b.

Due to multi-intermediate nature and diversity of chemical reactions inside a cell, the metabolic network cannot reflect all different aspects of the metabolism. It usually represents incomplete knowledge, firstly due to limitations of graph representation that needs determination of metabolites relevant to construction of edges and secondly because for the great part of living organisms only a small part of metabolic pathways has been discovered so far. The automatic creation of metabolic networks is difficult as biochemical knowledge is required to select proper set of metabolites and form relevant edges.

The metabolic network of an organism, even created on the basis of complete set of reactions, does not describe completely the state of entity. The network is a static part of description while metabolic fluxes of reactions changing in response to environmental stimuli reflect its dynamics. The enzymes gene expression levels determine active portions of metabolism, which can be incorporated into metabolic network using edge weights. The study of cell metabolism requires costly high-throughput experiments, that despite of indubitable advances in the field, still cannot provide complete view in a form of, e.g., list of all proteins present inside a cell or list of reactions occurring in a cell. This is due to finite sensitivity of experiments and inherent difficulty arising as exceptions from *gene* $\rightarrow$ *enzyme* $\rightarrow$ *reaction* one-to-one mapping.

Metabolic network can be constructed using single-organism pathways or all known reactions. In the latter case we obtain a super-metabolic-network that develops as far as new metabolic pathways are discovered, nevertheless is does not represent any living organism [97]. Determining comprehensive metabolic network for the given species can be impossible as metabolism of a few model organisms was explored profoundly. The collections of metabolic pathway data for many species is provided by databases `http://www.genome.jp/kegg` and `http://biocyc.org`.

The described network is directed, labeled graph that can form the basis for further investigations. The first issue that can be addressed is the global structure of metabolic networks. Do they vary among different species and to what extent the physiology of an organism is reflected by the network topology and dynamics? The study of metabolic networks revealed that generally they follow *scale-free* in-degree and out-degree distributions and posses *small-world* property [92, 155]. These findings hold for metabolic networks constructed in different ways. Nevertheless exceptions exist and due to incompleteness and limited range of data some results should be treated as rough approximation. The average path length depends on the type of metabolic network [97]. For certain types the values of 8.3 and 8.1 were reported, what is more than for random graph of similar size [16,125]. Besides, the metabolic networks of *Archaea*[1] appear to be more random-like and far from *scale-freeness* [169]. The question whether metabolic graphs epitomize complex networks community governed by *preferential attachment* rule or are

[1]Domain of single-celled microorganisms without cell nucleus, which can be found in extreme environments such as salt lakes or hot spots

the result of biologically and spatially constraint optimization is still under debate [97].

The clustering of network nodes confirmed functional subdivisions of metabolism such as *catabolism*[1] or *lipidic biosynthesis*. The modules are organized hierarchically, so that submodules combine into greater modules [89, 130]. The application of different hierarchical clustering algorithms brought more interesting insights such as revealing *giant connected component* representing inter-convertible metabolites and associated substrate and product subnetworks forming so-called *bow-tie* structure [109]. The degree distributions in separate modules are exponential, what in combination with highly connected *carrier metabolites* yields power-law global distribution [145]. The study of functional organization of metabolic graphs is a prominent application of graph community detection algorithms.

The next issue concerning metabolic networks is their robustness to damage caused by mutations or purposeful silencing of genes encoding enzymes. Owing to heavy-tailed degree distributions they appear to be resistant to random attacks and vulnerable to targeted attacks, what is a common feature of complex networks. However, this does not mean that metabolites of low connectivity are less important. Due to cascading perturbations caused by elimination of single enzyme (lack of certain product p may stop next reaction for which p is a substrate) the resulting damage cannot be modeled as simple deletion of single vertex. Contrary to findings derived from complex networks approach, the highest damage to metabolic network is caused by knocking-out enzymes that catalyze reactions producing metabolites of low degree [145].

The study of network vulnerability can be used in identifying targets for drugs [99, 167] e.g. looking for lethal disruption of bacteria metabolic networks, so that the same drug does not harm host organism. Such analysis requires recognition of essential edges (reactions) what can be obtained using edges descriptors [18]. Last research efforts focus on multi-target attack that was proven to be more effective than single-target approaches [14, 51]. Research carried out on metabolic data allows also for uncovering evolutionary relations between organisms. Such investigation uses hierarchical clustering algorithms based on substrate-product lists [122] or metabolic-network descriptors [17]. The resulting phylogenetic trees allows for uncovering gene-level similarity between organisms that cannot be obtained on phenotype-level.

3.4 Graphs in medicine

Medical applications of structured data include several types of graphs such as brain fuctional organization networks [26], networks modeling connections between neurons and gene co-expression networks used in complex disease gene mapping. Nevertheless, in this work we particularly focus on a structural representation of blood vasculature.

3.4.1 Vascular networks

Vascular networks deliver oxygen and nutrients to tissues, therefore modeling their formation and behavior plays significant role in understanding processes taking place in organs. Normal vasculature at capillary level resembles regular, planar mesh (see Figure 3.4a). This is because it evolved to provide approximately uniform spatial distribution of chemicals in surrounding tissue. After changing scale from micrometers to millimeters, we can observe hierarchically organized arterio-venous vessels forming tree-like structures with many three-way junctions. This

[1]Metabolic reactions which release energy by breaking down complex molecues into simpler ones, e.g., glycolysis

type of vessels provide long-range transport. The situation changes when vasculature in presence of tumor is examined (see Figure 3.4b). Here, the observed structures are heterogeneous, non-hierarchical and fragmented into zones of different microvascular density [162]. Contrary to normal vasculature, tumor blood vessels exhibit fractal geometry with many irregularities of different sizes [22]. The process of tumor growth is divided into five phases.

1. **Oncogenesis** - DNA reparation system disrupted by mutation causes the esacape of mutated cell from apoptosis. Normal cell transforms into tumor cell.

2. **Avascular growth** - supplied by diffusion-induced concentration of nutrients and oxygen, tumor cells start to proliferate, reaching certain maximum size determined by amount of food supply they acquire via diffusion.

3. **Angiogenesis** - due to unsufficient oxygen supply tumor cells come to hypoxia and start to produce tumor angiogenic factors, which diffuse through the tissue and stimulate growth of vasculature towards tumor.

4. **Vascular growth** - the tumor is connected to vascular network which provides practically unlimited access to resources and means of transport. After reaching this state the prognosis becomes poor as tumor cells can be transported to different parts of host organism and form metastases.

5. **Metastasis** - metastatic tumos appear in other organs.

Certainly, the transition from state 3 to 4 is critical, therefore many efforts were made to elucidate complex dynamics of tumor vasculature growth and investigate how its structure influences tumor resistance to drugs [148, 161]. Quantitative analysis of vascular networks topology has considerable impact on advances of anti-angiogenesis therapy in cancer. Comparing blood vessels in different metastatic tumors of the same type may test hypothesis about their similarity. Promising application of graph matching algorithms arises as far as inter-regional variation of vasculature in tumor is considered. Investigating relations between vascular networks in different types of tumors can also bring valuable conclusions.

In the first approach to the study of tumor blood vessels, spatial parameters such as microvascular density (MVD) or box fractal dimension [22] were used. Unfortunately, the dependence of those measures on the spatial position of sample make them not universal. This is where purely topological network descriptors can be more helpful. Following core findings from theory of complex networks [30], structural properties of vasculature such as *connectivity*, *modularity* or *scale-freeness* can reflect functions and qualitative features of underlying complex system. The information encoded by the structure of a network is extracted using dedicated descriptors. For instance, *edge betweeness centrality* or *clustering coefficient* enables to find sub-networks playing the most significant role in the whole system and allows for grouping functionally similar nodes. Graph descriptors can be also used to keep track of tumor vasculature dynamics and comparison of networks at different stages of development.

Regardless of being a promising research subject, digitalized vascular networks are expensive and troublesome to obtain. Images from angiography or confocal microscopy can be segmented and after skeletonization transformed to adjacency matrices (see Figure 3.4c,d), however such a two-dimensional approach is definitely not perfect as vasculature is most likely a complex 3D structure. Two-dimensional real-world vascular networks are also obtained by special tumor inoculation into laboratory mice [74] but the cost of such a procedure is prohibitively

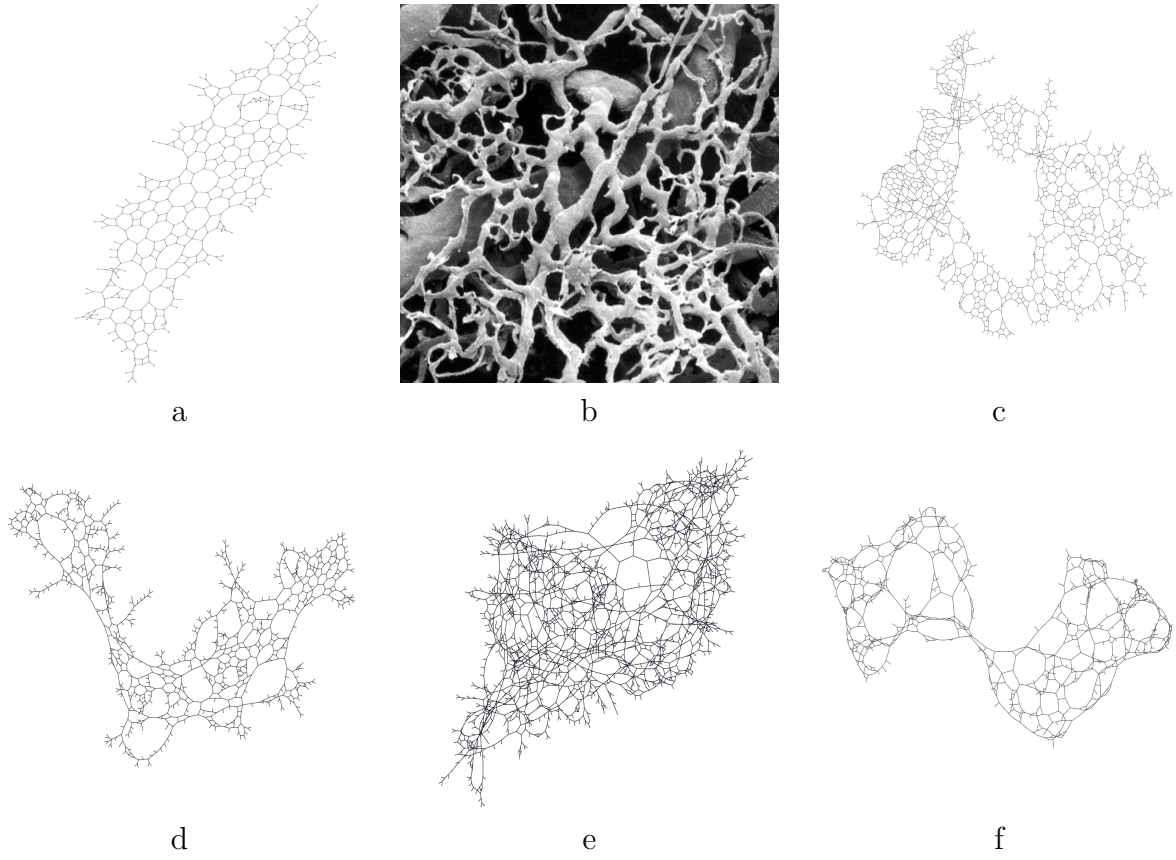


Figure 3.4: Samples of vascular networks (a) normal subcutaneous capilary network [74], (b) confocal microscopy image of tumor microvasculature [112], (c)(d) 2D tumor vasculature obtained from tumor-inoculated mice [22, 74], (e) vascular network generated by discrete-continuum model [161], (f) vascular network generated by tumor growth model based on cellular automata [148]. All networks visualized on the basis of purely structural representation using *Graph Investigator* application.

big. Therefore, *in-silico* experiments which can deliver vast amount of simulation data, play prominent role in providing new insights into tumor growth mechanisms. Graph comparison algorithms are applied here to validate the results of simulation with a real-world data [160].

In recent years many competing models of tumor-induced angiogenesis were created. Among them, hybrid 3D model which combines particle dynamics, cellular automata and continuum approach [161] appears to be most advanced. The sample tumor vasculature obtained from simulation based on this model is depicted in Figure 3.4e. In turn, the model presented in work [148] takes into consideration newly discovered angiogenic factors and includes remodeling of existing vasculature. It employs graph of cellular automata as a substrate for tumor and vasculature growth. The sample output of this model is presented in Figure 3.4f. The *Graph Investigator* application descibed in Chapter 5 was developed as a validation tool for those two modeling frameworks.

3.5 Graphs in other disciplines

For the sake of completeness we briefly review sample graph data in other disciplines. Thereby, the explanation of diagram presented in Figure 1.2 is provided.

- **Sociology**

  The study of social networks which model various relations and associacions between individuals has a long history. Many works on complex networks used this type of data for analysis [24, 25, 30]. This includes graphs reflecting friendship, coauthorship, email or telephone contacts, sexual activity or business relations. In today's Facebook era, social network meme is also recognized in a popular context.

- **Chemistry**

  Chemical compounds are represented by attributed graphs with edges modeling bonds. Predicting anti-cancer activity, mutagenicity or toxicity is one of problems set by those data [126].

- **Physics**

  The folding process in which long amino-acid sequences transform into three-dimensional tertiary protein structures can be encoded as a graph with edges modeling transitions between configurations which are represented by vertices. Quantitative analysis of such a model can bring new insights into understanding of folding kinetics [41, 76, 108].

Part II

Contribution

Chapter 4

Invariants of Distance k-Graphs for Graph Embedding

In this chapter we explore graph embeddings constructed on the basis of invariants derived from distance k-graphs [37], which are built upon the information about shortest paths of a given graph. As a set of distance k-graphs is ordered, it constitutes a good basis for extracting features invariant under graph isomorphism. We extend the definition of distance k-graphs to edge distance k-graphs, which store more information about graph structure. Next, we show how by collecting invariants of successive distance k-graphs, one can generate robust graph feature vectors, efficient in clustering and classification of structural patterns. Particularly, we focus on distance k-graphs degree distributions, which form graph B-matrix representations [20, 57] and can be obtained with relatively low computational cost $\mathcal{O}(n^2)$. Moreover, recent developments in GPU implementations of classical graph algorithms such as all-pair shortest-paths (APSP) or breadth-first-search (BFS) allow for increasing the size of graphs analyzed interactively using distance information by two orders of magnitude [86]. This motivated us to make use of CUDA implementation of R-Kleene APSP algorithm [38] for computation of distance matrices and generation of distance k-graphs for large networks. Therefore graph descriptors presented in this work are an interesting alternative for spectral methods that usually require $\mathcal{O}(n^3)$ steps. In an experimental setting, by testing embedding stability and classification rates on real-world datasets, our feature vectors are compared with graph characteristics derived from a heat kernel matrix.

The outline of this chapter is as follows. First, in section 4.1.1 we recall definition of distance k-graphs and show that their invariants can be used to distinguish graphs from which they were derived. In section 4.1.2 we describe vertex B-matrix which is constructed on the basis of distance k-graphs degree distributions. The next sections introduce edge distance k-graphs which encode more information about graph structure and define edge B-matrix invariant. In section 4.2 we demonstrate how B-matrices can be used for graph embedding. Later, experiments on artificial and real-word data are presented including embedding stability test, unsupervised and supervised graph learning. In the end we draw conclusions and discuss ideas for further development of this approach.

4.1 Distance k-graphs and B-matrices

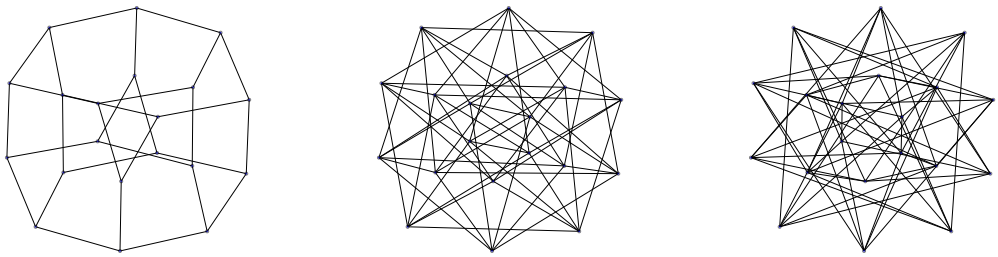
In this section we show how the information about shortest paths in a graph G can be exploited for generating ordered set of G-related graphs.

4.1.1 Vertex distance k-graphs

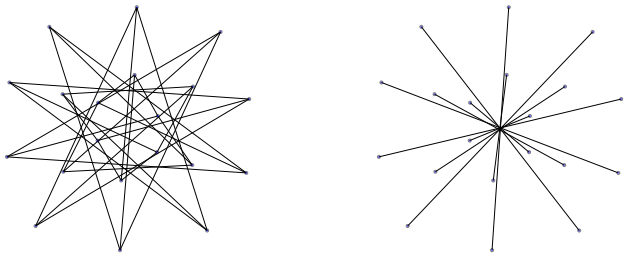
To commence we recall definition of distance k-graph [37].

Definition 4.1.1 *For a graph $G = (V(G), E(G))$ we define distance k-graph $G_k^{\mathcal{V}}$ as a graph with vertex set $V(G_k^{\mathcal{V}}) = V(G)$ and edge set $E(G_k^{\mathcal{V}})$ so that $\{u, v\} \in E(G_k^{\mathcal{V}})$ iff $d_G(u, v) = k$.*

If $d_G(u, v) = k$ the vertices u and v are called k-neighbors. From this definition it follows that $G_1^{\mathcal{V}} = G$ and for $k > diam(G)$ $G_k^{\mathcal{V}}$ is empty graph. The sample set of distance k-graphs for Desargues graph is depicted in Figure 4.1.



a. $k = 1$, 30 edges b. $k = 2$, 60 edges c. $k = 3$, 60 edges

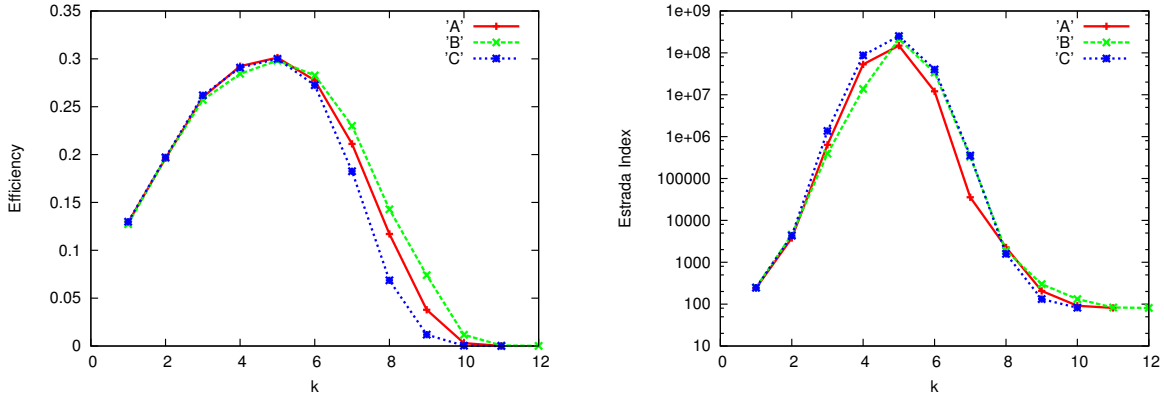


d. $k = 4$, 30 edges e. $k = 5$, 10 edges

Figure 4.1: Distance k-graphs for Desargues graph (20 edges, 30 vertices, diameter 5).

With growing value of k, the information about connectivity of G encoded in $G_k^{\mathcal{V}}$ graphs moves from local to global. Several observations can be made after looking at Figure 4.1. We recognize that the density of distance k-graphs changes non-monotonically with k. First, it grows and after reaching certain midpoint, it begins to decrease. This 'turning point', occuring in $\{G_k^{\mathcal{V}}\}$ sets of many graphs, is related to finite size effects. After k passes average shortest path length, the number of vertices separated by distance k decays. The $G_k^{\mathcal{V}}$ graphs are not necessarily connected. For Desargues graph shown in Figure 4.1a, $G_1^{\mathcal{V}}$ and $G_3^{\mathcal{V}}$ are connected, while $G_2^{\mathcal{V}}$, $G_4^{\mathcal{V}}$ and $G_5^{\mathcal{V}}$ are not connected.

The questions how the structure of G determines topological properties of $G_k^{\mathcal{V}}$ graphs and whether G can be univocally reconstructed from the $\{G_k^{\mathcal{V}}\}$ set has been studied in graph theory literature from the 70s [37]. For instance, in [151] it was shown what is the maximal number of k-neighbors for a given graph G. The number of edges in distance k-graphs is maximized for so-called *t-broom* graphs. Furthermore, coloring of distance k-graphs finds applications in efficient computation of Hessians [75]. The k-th power of a graph, denoted by G^k is a transformation similar to forming distance k-graphs. Here, the edge between vertices u and v is created iff $d_G(u, v) \leqslant k$ [140].



a b

Figure 4.2: Efficiency (A.1) (a) and Estrada Index (A.11) (b) of distance k-graphs for three random connected Erdős-Rényi graphs (80 vertices, 100 edges). The graphs possess nearly the same values of $G_1^{\mathcal{V}}$ efficiency, the difference occurs for $k > 6$ when higher-level features are taken into account. Also Estrada Indexes of $G_1^{\mathcal{V}}$ graphs are close, nevertheless the subsequent distance k-graphs bring more discriminative information ($k = 4$, $k = 6$, $k = 7$).

In order to evaluate discriminative abilities of distance k-graphs invariants we generated three structurally similar random Erdős-Rényi graphs (80 vertices, 100 edges) and computed values of efficiency and Estrada Index for $\{G_k^{\mathcal{V}}\}$ sets. The Estrada descriptor, defined as a trace of the exponential adjacency matrix, quantifies the content of subgraphs in the graph [63]. As presented in Figure 4.2, the higher-level features encoded by distance k-graphs, allow for capturing subtle structural differences between random graphs of the same density. By enumerating scalar graph descriptors of $G_k^{\mathcal{V}}$ graphs we can obtain feature vector of maximum length $diam(G)$. For dense graphs this gives a low-dimensional representation that can be not discriminative enough. In order to extract more information from the structure of distance k-graphs we plan to use constant-bin histograms describing frequencies of certain element descriptors like vertex degree.

4.1.2 Vertex B-matrix

Vertex degree frequencies store information about local neighborhood of distance k-graphs. From the perspective of source graph G, the locality of this information moves towards globality with increasing k. In this section we construct permutation invariant representation of a graph, composed of histograms of $G_k^{\mathcal{V}}$ graphs degree frequencies, that is also known as vertex B-matrix [20, 57].

Definition 4.1.2 *We define k-shell of vertex v as a subset of graph vertices at distance k from v (k neighborhood of vertex v).*

The sample vertex 2-shell is depicted in Figure 4.3a. Degree of order k is a size of respective k-shell of vertex v. The nearest neighbors of v form its 1-shell and the size of this 1-shell is degree of vertex v.

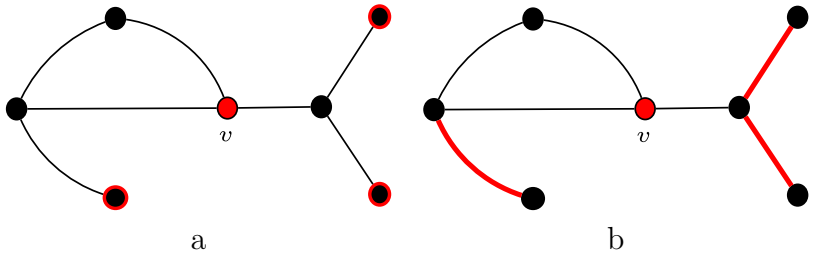


Figure 4.3: a. Sample vertex 2-shell of vertex v, b. sample edge 1.5-shell of vertex v.

Definition 4.1.3 *The vertex B-matrix of a graph with n vertices is defined as follows.*
$B^{\mathcal{V}}_{k,l}$ *= number of nodes that have l members in their k-shells,*
where $k \leq diam(G)$ *and* $l \leq n$.

From the definition above it follows immediately that $B^{\mathcal{V}}_{k,l}$ is a number of vertices with degree l in a respective $G^{\mathcal{V}}_k$ graph. The edge in $G^{\mathcal{V}}_k$ joins two vertices u and v iff $d_G(u,v) = k$. This is equivalent to the fact that u belongs to k-shell of v and vice versa. By counting vertices incident to u in $G^{\mathcal{V}}_k$ we establish the size of k-shell of u. The example of vertex B-matrix for brain valcular network with 1090 vertices and 1438 edges is presented in Figure 4.4b.

To enumerate shell members for each vertex of the graph we use Breadth-First Search (BFS) algorithm which runs with $\mathcal{O}(n+m)$ steps, where n is the number of graph vertices and m is the number of edges. Therefore for sparse graph, vertex B-matrix can be generated with computational cost $\mathcal{O}(n^2)$ and for dense ones with $\mathcal{O}(n^3)$ time complexity.

The structure of vertex B-matrix reflects the structure of an underlying graph, hence $B^{\mathcal{V}}$ can be used for visual comparison and classification of various networks. It is capable of capturing such high-level properties of networks as assortativity/disassortativity, small-worldliness or regularity [20]. The level of histogram overlap between consecutive rows of $B^{\mathcal{V}}$ indicate network branching. For dense, well-connected graphs the size of k-shells grows rapidly with k so that neighboring histograms are far apart (see Figure 4.5a). The opposite behavior is observed for sparse graphs (Figure 4.5b), with a special case of a path graph possessing only two non-zero columns in $B^{\mathcal{V}}$ (Figure 4.5c). The broadness of $G^{\mathcal{V}}_k$ vertex degree distributions indicate graph regularity. Here, regular and distance-regular graphs give single non-zero entry in each row, while highly irregular graphs yield wide, uniform-like histograms, especially for k close to average shortest path length. The 'turning point' described in the previous section is also present in many vertex B-matrices (see Figure 4.4b). In case of not-connected graph G (infinite diameter), only connected pairs of vertices are considered, so that resulting $B^{\mathcal{V}}$ matrix is a sum of $B^{\mathcal{V}}$ matrices for all connected components of G.

Despite its robustness in encoding information about graph, vertex B-matrix is not a complete graph invariant. This means that there exist pairs of non-isomorphic graphs possessing the same vertex B-matrix representation. The example of such a pair is dodecahedral and

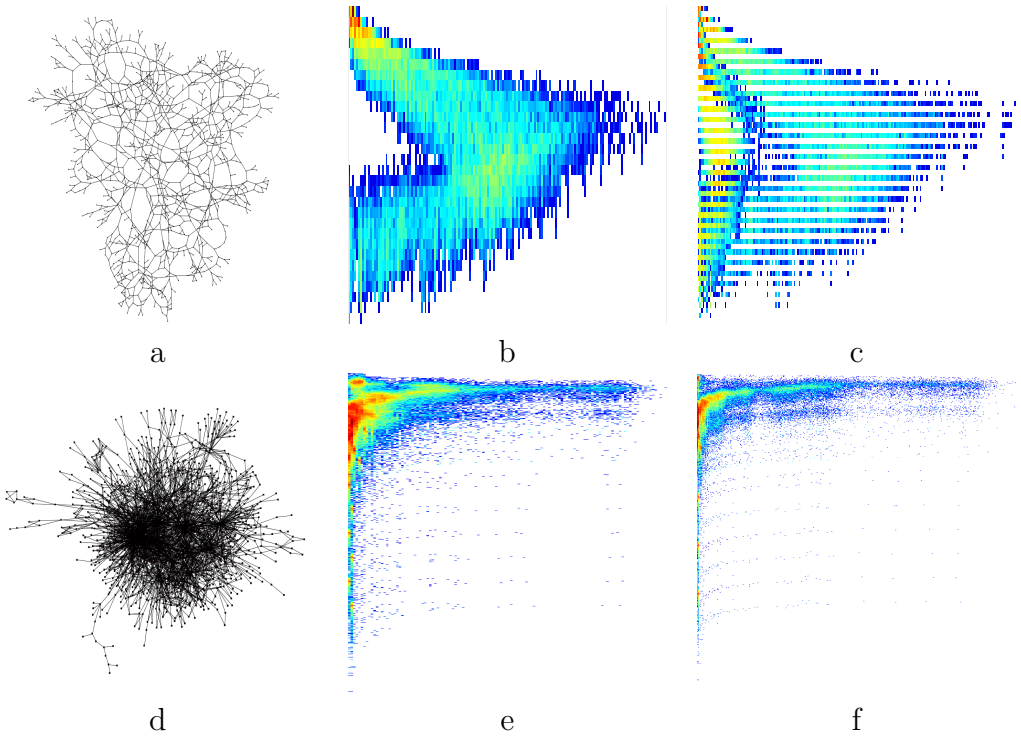


Figure 4.4: a. Brain vascular network G with 1090 vertices, 1438 edges and diameter 30, b. Vertex B-matrix of G, c. Edge B-matrix of G, d. Protein-protein interaction network H of *Aquifex aeolicus* (1473 vertices, 3354 edges), e. Vertex B-matrix of H derived from commute time metric, f. Edge B-matrix of H derived from commute time metric. Logarithmic color scale is used.

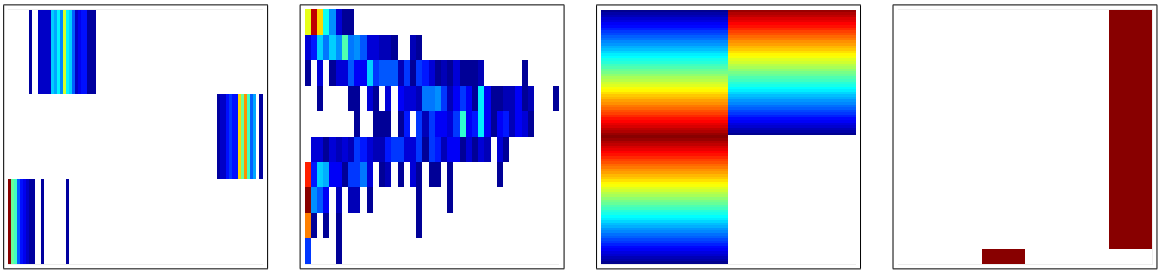


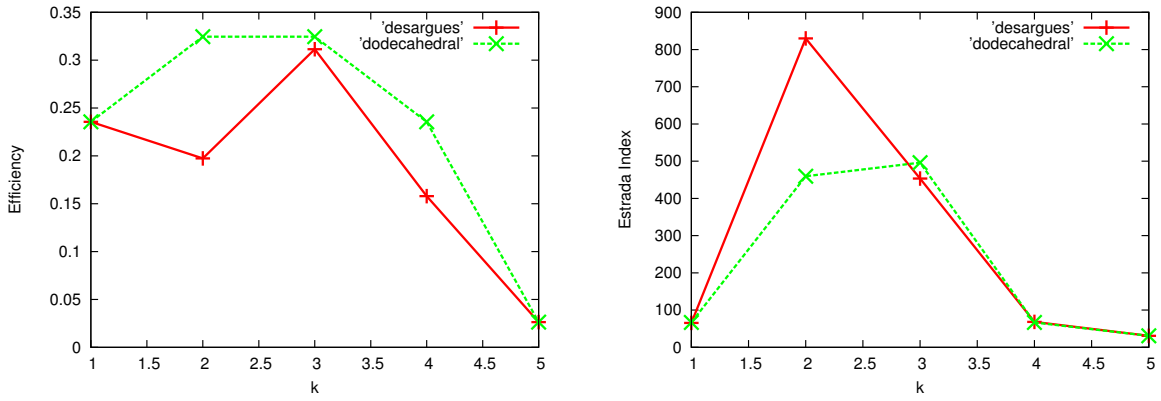
Figure 4.5: Examples of vertex B-matrices for graphs with 100 vertices: a. dense Erdős-Rényi graph ($p = 0.01$), b. sparse Erdős-Rényi graph ($p = 0.001$), c. path graph, d. 6-regular graph

Desargues graphs. Nonetheless, the sets of distance k-graphs for these two graphs are distinct and they can be distinguished using different invariants such as efficiency or Estrada Index (see Figure 4.6).

4.1.3 Edge distance k-graphs

We start the presentation of edge distance k-graphs with a definition of vertex-edge distance.

Definition 4.1.4 *Let $G = (V(G), E(G))$ be an undirected, unweighted, simple graph. The distance from a vertex $w \in V(G)$ to an edge $e_{uv} = \{u, v\} \in E(G)$, denoted as $d_G^{\mathcal{E}}(w, e_{uv})$, is the*



a b

Figure 4.6: Efficiency (a) and Estrada Index (b) of distance k-graphs for Desargues and dodecahedral graphs that posses the same vertex B-matrix.

mean of distances $d_G(w,u)$ *and* $d_G(w,v)$.

In case of unweighted graphs such dissimilarity measure has integer or half-integer values.

Definition 4.1.5 *We define edge distance k-graph as a bipartite graph* $G_k^{\mathcal{E}} = (U(G_k^{\mathcal{E}}), V(G_k^{\mathcal{E}}), E(G_k^{\mathcal{E}})) = (V(G), E(G), E(G_k^{\mathcal{E}}))$ *such that for each* $w \in V(G)$ *and* $e_{uv} \in E(G)$, $\{w, e_{uv}\} \in E(G_k^{\mathcal{E}})$ *iff* $d_G^{\mathcal{E}}(w, e_{uv}) = k$.

In Figure 4.7 the sample undirected graph is presented together with its edge distance 1-graph (Figure 4.7b) and edge distance 1.5-graph (Figure 4.7c).

For a given vertex w, integer values of k account for edges e_{uv} whose endpoints are equidistant from w. This means that e_{uv} belongs to odd closed walk of length $2k+1$ starting and ending at w. For instance, edge distance 1-graph shown in Figure 4.7b contains information about triangles in the graph G. Each triangle is encoded tree times by joining a vertex with the triangle-opposite edge, e.g., for triangle $\{3,4,5\}$ vertex 3 is connected to edge $\{4,5\}$, 4 to $\{3,5\}$ and 5 to $\{4,3\}$. In a similar way $G_2^{\mathcal{E}}$ stores information about closed walks of length 5, etc. On the other hand half-integer values of k account for edges e_{uv} so that $|d_G(w,u) - d_G(w,v)| = 1$. This represents closed walks of even length $2k+1$ (k is non integer this time) which start and end at w. For instance, edge 0.5-distance graph is a bipartite graph joining vertices of G with incident edges. The maximal value of k for which $G_k^{\mathcal{E}}$ is non-empty is $2 \times diam(G)$. Moreover, in case of bipartite graphs, that do not posses odd cycles and for $k = 1, 2, 3, \ldots, 2 \times diam(G)$ $G_k^{\mathcal{E}}$ is an empty graph.

The size of edge distance k-graphs is greater than distance k-graphs, therefore we expect them to be more information rich. In order to illustrate the utility of $G_k^{\mathcal{E}}$ invariants in distinguishing graphs we computed $G_k^{\mathcal{E}}$ efficiency descriptors for three random graphs used previously in section 4.1.1. This time edge distance k-graphs were divided into two groups according to value of k (integer or not-integer). As shown in Figure 4.8a for non-integer values of k we obtained results consistent with those received for $G_k^{\mathcal{V}}$ (see Figure 4.2a). The values of efficiency for $k \geq 6.5$ vary from one another. Interestingly, integer k's (odd-length closed walks features) bring additional discriminative information (see Figure 4.8b). Here the distinction is clearer, even for low values of k.

As computation of efficiency or Estrada Index for $G_k^{\mathcal{E}}$ requires additional computational overhead we plan to explore computationally less-expensive invariants such as degree distributions of vertices from $U(G_k^{\mathcal{E}})$ or $V(G_k^{\mathcal{E}})$ that can be organized in a form of edge B-matrix.

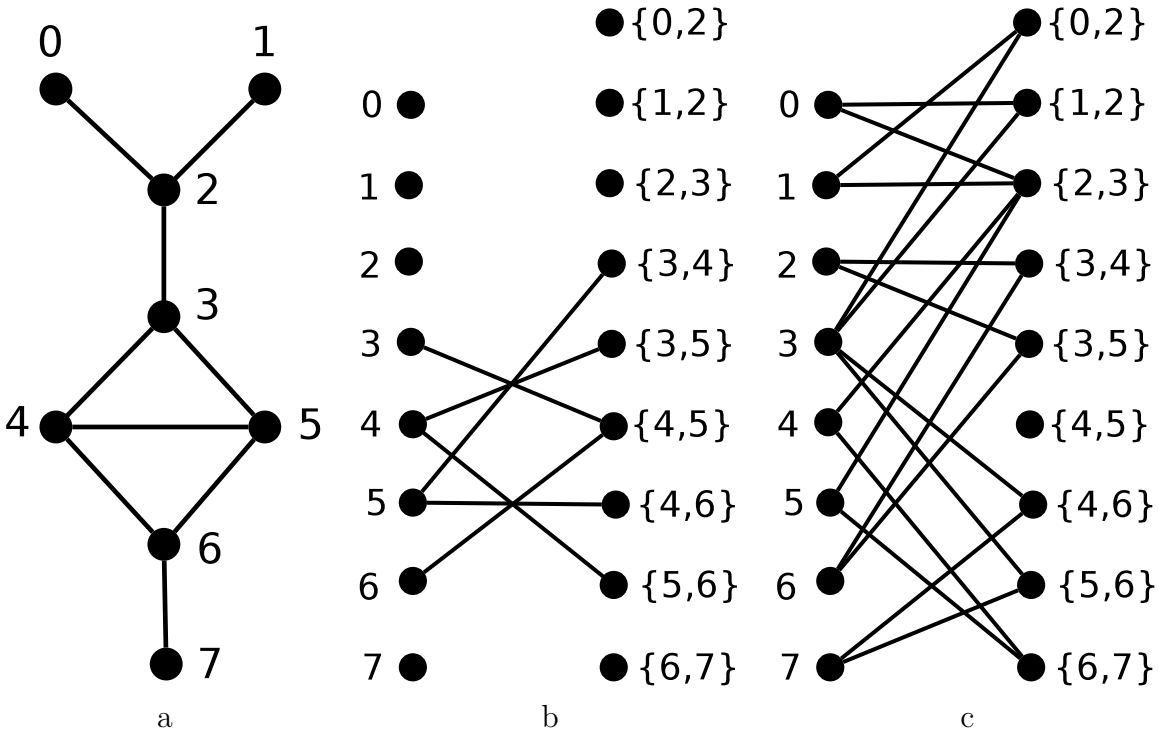


Figure 4.7: a. Connected graph G, b. Edge distance 1-graph derived from G, c. Edge distance 1.5-graph derived from G.

4.1.4 Edge B-matrix

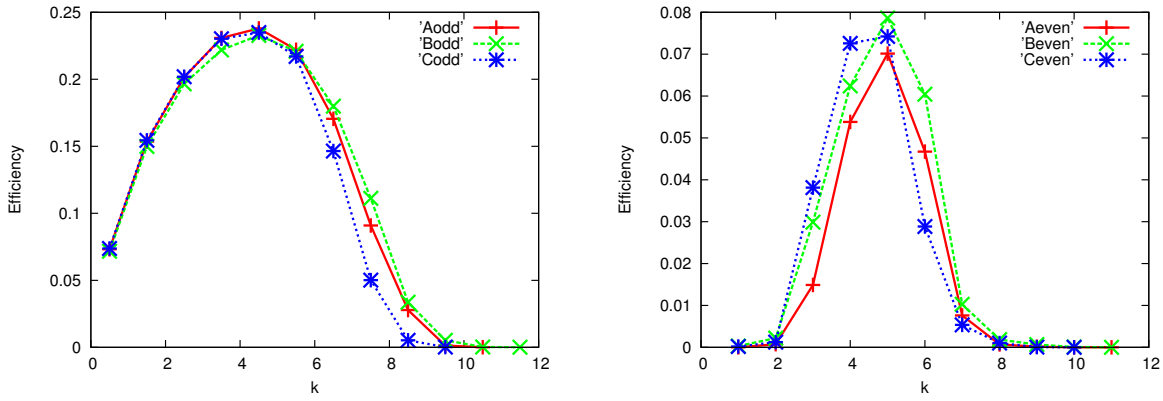
In order to introduce edge B-matrix we employ the notion of vertex-edge distance defined in the previous setion.

Definition 4.1.6 *The k-edge-shell of vertex v is a subset of graph edges at distance k from v (k can have half-integer values)*

The sample 1.5-edge shell of a given vertex is depicted in Figure 4.3b.

Definition 4.1.7 *The following equation defines edge B-matrix of a graph.*
$B_{i,l}^{\mathcal{E}}$ *= number of nodes that have l edges in their* $(\frac{1}{2}i)$*-edge-shells.*

In other words i-th row of $B^{\mathcal{E}}$ stores degree frequencies of vertices from $U(G_{0.5i}^{\mathcal{E}})$ set of bipartite $G_{0.5i}^{\mathcal{E}}$ graph. The vertices from $U(G_{0.5i}^{\mathcal{E}})$ are connected with edges belonging to their edge $0.5i$-shell, therefore the number of vertices with degree l in $U(G_{0.5i}^{\mathcal{E}})$ equals $B_{i,l}^{\mathcal{E}}$. The maximum size of $B^{\mathcal{E}}$ is $(2 \times diam(G)) \times m$, where n denotes number of vertices and m number of edges. The sample edge B-matrix of brain vascular network is depicted in Figure 4.4c. In order to construct $B^{\mathcal{E}}$ matrix, we need to have vertex-vertex distance information. The computational cost of obtaining it is $\mathcal{O}(n^3)$ for dense and $\mathcal{O}(n^2)$ for sparse graphs (the same as for vertex



a b

Figure 4.8: Efficiency of edge distance k-graphs for three random connected Erdős-Rényi graphs (80 vertices, 100 edges) a. non-integer values of k, b. integer values of k.

B-matrix). The difference is that this time we need more memory, as the number of columns in $B^{\mathcal{E}}$ depends on edge count. For dense graphs this becomes computationally cumbersome, as the number of columns reaches the order of n^2. Nevertheless, owing to narrow row distributions, after removing zero columns, effective size can be decreased significantly.

Following remarks from section 4.1.3, we can establish several correspondences between structural properties of G and shape of its edge B-matrix. First, for graphs without odd cycles (bipartite graphs) even rows of $B^{\mathcal{E}}$ are empty. This property cannot be easily derived either from typical graph representations or from graph planar embeddings. Triangle-free graphs possess empty second row ($k = 1$). Graphs without triangles and pentagons have empty sixth row ($k = 3$). This is because closed walks of length 5 are pentagons, triangles with two-times traversed edge attached to one of triangle vertices or triangles with one edge traversed three times.

Edge B-matrices are different for Desargues and dodecahedral graphs that possess identical vertex B-matrices. This prompted the authors of [20] to conjecture that this representation together with $B^{\mathcal{V}}$ forms a complete graph invariant. The positive answer to this question would be equivalent to proving that graph isomorphism problem belongs to **P** complexity class.

4.1.5 Weighted graphs

The approach presented in previous sections can be extended to weighted graphs or different vertex-vertex metrics, e.g., commute time [123] (see Appendix D). Assuming real non-negative values of d_G we redefine distance k-graphs and edge distance k-graphs as follows. Let r be the number of distance bins[1] and let $b = diam(G)/r$ denote distance-bin size.

Definition 4.1.8 *The distance k-graph of graph $G = (V(G), E(G))$ is as a graph $G_k^{\mathcal{V}} = (V(G_k^{\mathcal{V}}), E(G_k^{\mathcal{V}}))$ so that $V(G_k^{\mathcal{V}}) = V(G)$ and $\{u, v\} \in E(G_k^{\mathcal{V}})$ iff $(k-1)b < d_G(u, v) \leqslant kb$, whereby $k = 1, 2, \ldots, r+1$.*

Correspondingly, for real non-negative values of $d_G^{\mathcal{E}}$ we define edge distance k-graph.

[1]The maximum value of d_G is divided into r equal intervals

Definition 4.1.9 *The edge distance k-graph of graph* $G = (V(G), E(G))$ *is a bipartite graph* $G_k^{\mathcal{E}} = (U(G_k^{\mathcal{E}}), V(G_k^{\mathcal{E}}), E(G_k^{\mathcal{E}})) = (V(G), E(G), E(G_k^{\mathcal{E}}))$ *such that for each* $w \in V(G)$ *and* $e_{uv} \in E(G)$, $\{w, e_{uv}\} \in E(G_k^{\mathcal{E}})$ *iff* $(k-1)b < d_G^{\mathcal{E}}(w, e_{uv}) \leqslant kb$, *where* $k = 1, 2, \ldots, r+1$.

By adjusting value of r we get fine-grained or coarse-grained distance k representations of a source graph G. These graphs form the basis for graph embedding. For instance, B-matrices derived from commute time metrics computed for protein-protein interaction network of *Aquifex aeolicus* bacteria are depicted in Figure 4.4e,f.

4.1.6 Shortest paths algorithms on GPU

Computation of graph feature vectors becomes infeasible for larger graphs. With a typical time complexity $\mathcal{O}(n^3)$ or at least $\mathcal{O}(n^2)$, the significant part of graph embedding algorithms suffer from data size-dependent long processing times. This problem can be tackled with massively parallel architectures of modern GPUs and particularly using CUDA programming model. In this section we would like to present several remarks on recently developed GPU implementations of all-pair shortest-paths (APSP) and breadth-first-search (BFS) algorithms, which can bring significiant performance boost to graph comparison based on distance k-graphs and B-matrices.

Highly optimized GPU-enabled version of APSP recursive Kleene algorithm was described in [38]. It uses parallelized, *in-place* version of fast matrix multiplication routines [154] and stores whole distance matrix in the device memory. The key concept, that lies under its efficiency, is that the problem of computing all-pair shortest-paths is rendered as a set of recursive matrix-matrix multiplications on a tropical closed semiring [60] (see Section 6.3.1). Owing to the idempotence of *min* operation, the modified matrix multiplication (*min* and $+$ instead of $+$ and $*$) can be performed *in place* as the threads from different blocks are allowed to modify the same global memory area without violating validity of the solution. Recursive formulation of APSP increases data locality, which together with a usage of fast shared memory, yields outstanding, two-orders of magnitude speedup over CPU implementations [38, 59]. For instance, on Nvidia Tesla C2070, which possesses 448 multiprocessors and 6GB memory, the computation of distance matrix for a random graph with 4196 vertices and 16863 edges takes less than 1 second (including host-to-device memory transfers). During computation the distance matrix is stored in GPU memory using single precision. This limits the size of graphs that can be processed, e.g. for previously mentioned Nvidia Tesla C2070 the margin is $4 \cdot 10^4$ vertices. For larger graphs, Buluç R-Kleene implementation becomes infeasible.

The use of condensed adjacency lists as a graph representation and CUDA implementation of BFS algorithm allows for moving GPU memory size limit a step forward [86]. Here, the shortest paths are computed separately for each vertex without the need to store whole distance matrix in the device memory. In the bulk-synchronous-type BFS implementation presented in [86], each thread is attached to a single vertex and level-synchronization together with vertices-activity arrays are used to traverse graph. Random-type access to adjacency lists cannot be coalesced, therefore, the speedup of BFS GPU implementation is much lower that one obtained for APSP GPU algorithm and ranges from 2 to 6, depending on graph density.

In this work, we used optimized GPU implementation of R-Kleene algorithm integrated with *Graph Investigator* application to generate distance k-graphs of larger networks and compute their B-matrices [58, 59]. The details of this part of research are explained in Chapter 6.

4.2 Graph descriptors from distance k-graphs

In this section we define graph embeddings based on distance k-graphs invariants, particularly on graph B-matrices. Let $\star$ stand for $\mathcal{V}$ or $\mathcal{E}$ symbol, so that $B^\star$ denotes $B^\mathcal{V}$ or $B^\mathcal{E}$ matrix of a graph G.

The most straightforward way of graph embedding is packing rows or columns of B-matrices to form a long pattern vector.

$$\begin{aligned} &D^\star_{long}(k_{min}, k_{max}, l_{min}, l_{max}) = [B^\star_{k,l}] \\ &k_{min} \leq k \leq k_{max},\ l_{min} \leq l \leq l_{max} \end{aligned} \tag{4.1}$$

Such an approach allows for retaining all the information present in a B-matrix. Using parameters k_{min}, k_{max}, l_{min} and l_{max} one can extract the B-submatrix that combines selected subset of low and high-level features. For instance by choosing $1 \leq k \leq 2$ and $1 \leq l \leq n$, where n is number of graph vertices, we extract local information about nearest and second-nearest-neighbors. In turn, after setting $1 \leq k \leq diam(G)$ and $1 \leq l \leq 3$, $D^\star_{long}$ stores information about k-shells with a small number of elements, i.e., counts vertices that are relatively distinct from 'typical' ones either because of their location in the center of the graph, or due to low local branching factor. The high-dimensional $D^\star_{long}$ descriptor forms a good basis for further feature selection (see Section 4.3.5).

In case of graphs with different number of vertices or different diameters, B-matrices have to be scaled by padding zero-rows or columns to obtain pattern vectors of the same length. In order to reduce dimensionality of $D^\star_{long}$, the size of k-level degree histogram bins can be increased. This reduces the number of columns in B-matrix. Besides, the number of rows can be decreased using a smaller number of distance-bins as described in section 4.1.5 for general distance k-graphs. In this manner more coarse-grained B-matrices are generated. Logarithmic scaling of non-zero entries of $B^\star$ allows for reducing dynamic range effects, making heavy-tailed degree distributions more tractable. Due to non-uniform structure of B-matrices, $D^\star_{long}$ possesses many zero elements, which for unsupervised learning can be easily removed from a dataset in the preprocessing step. In case of supervised task, the same subset shall be removed from a training and testing set.

As consecutive rows of B-matrices describe distributions of respective k-shell sizes, we use aggregated statistics to generate per-row characteristics and combine them into one feature vector. This can be particularly useful if the size of $D^\star_{long}$ is too large and lower-dimensional representation is needed.

$$D^\star_{rstd}(k) = \frac{\sigma^\star(k)}{\mu^\star(k)} \tag{4.2}$$

$D^\star_{rstd}(k)$ is relative deviation of row k, whereby $\mu^\star(k)$ denotes average and $\sigma^\star(k)$ standard deviation of k-shell size (only nonzero entries of $B^\star$ are taken into account). High values are achieved for left-skewed broad k-shell size distributions while low values indicate right-skewed narrow distributions. Therefore $D^\star_{rstd}(k)$ can be used as a measure of k-shell regularity.

The Shannon entropy $D^\star_{ent}(k)$ measures unpredictability of k-shell size. Low values of this descriptor indicate that k-shells of certain size dominate, whereas higher values are obtained for broad, uniform k-order degree distributions.

$$p(k,l) = \frac{B^\star_{k,l}}{\sum_l B^\star_{k,l}} \tag{4.3}$$

$$D^{\star}_{ent}(k) = -\sum_{l} p(k,l)\log(p(k,l)) \tag{4.4}$$

Assessing B-matrices inter-row diversity yields relevant graph features. The difference of average $(k-1)$-shell size and mean k-shell size reflects mean offset between consecutive distributions.

$$D^{\star}_{avgd}(k) = \mu^{\star}(k-1) - \mu^{\star}(k) \tag{4.5}$$

Negative values of $D^{\star}_{avgd}(k)$ indicate that k-shells have on average more members than $(k-1)$-shells. Large absolute values are obtained for dense graphs while small ones for sparse graphs. The change from negative to positive value of $D^{\star}_{avgd}(k)$ reflects the 'turning point' described in section 4.1.1.

For the sake of completeness, we also define distance k-graph efficiency and Estrada Index that were previously used in sections 4.1.1 and 4.1.3 to present discrimination abilities of distance k-graph invariants.

$$D^{\star}_{ef}(k) = ef(G^{\star}_{k}) \tag{4.6}$$

$$D^{\star}_{Est}(k) = Est(G^{\star}_{k}) \tag{4.7}$$

In case of edge distance k-graphs parameter k can have half-integer values. It is also interesting to note that the number of edges in $G^{\star}_{k}$ graph can be determined from respective B-matrix

$$|E(G^{\star}_{k})| = \frac{1}{2}\alpha\sum_{l} lB^{\star}_{\alpha k,l}, \tag{4.8}$$

where for $\alpha = 1$ for $B^{\mathcal{V}}$ and $\alpha = 2$ for $B^{\mathcal{E}}$. Let $\hat{V}(G)$ be a subset of $V(G)$ containing vertices with non-zero degree. It can be shown that

$$|\hat{V}(G^{\mathcal{V}}_{k})| = \sum_{l} B^{\mathcal{V}}_{k,l} \tag{4.9}$$

and

$$|\hat{U}(G^{\mathcal{E}}_{k})| = \sum_{l} B^{\mathcal{E}}_{2k,l}. \tag{4.10}$$

4.2.1 Related descriptors and transformations

Our method of graph embedding relies on the conversion of a source graph into sequence of derived graphs. Similar, transformation-based approach is utilized in the computation of Ihara coefficients [48, 131]. Here, the analyzed graph G is transformed into oriented-line Hashimoto graph encoding edge adjacency structure of G. The coefficients of the quasi characteristic polynomial of the adjacency matrix of this graph $\{c_1, c_2, \ldots, c_{2m}\}$ form the pattern vector reflecting prime cycle frequencies in the graph G. Particularly, c_3 coefficient being the negative of twice the number of triangles is closely related to the second row of $B^{\mathcal{E}}$ matrix. Let us denote the number of triangles in graph G as $n_{\triangle}(G)$. As indicated in Section 4.1.3, the graph

$G_1^{\mathcal{E}}$ conveys triangle structure of G, so that number of edges in $G_1^{\mathcal{E}}$ is three times $n_{\triangle}(G)$. Using edge B-matrix, it can be formulated as below

$$n_{\triangle}(G) = \frac{1}{3}|E(G_1^{\mathcal{E}})| = \frac{1}{3}\sum_l lB_{2,l}^{\mathcal{E}}. \tag{4.11}$$

In turn, descriptor $\mu^{\mathcal{E}}(2)$ measures average degree of $G_1^{\mathcal{E}}$ graph, taking into account only vertices with non-zero degree, that is vertices from the set $\hat{U}(G_1^{\mathcal{E}})$. Then, after noting that

$$\mu^{\mathcal{E}}(2) = \frac{2\sum_l lB_{2,l}^{\mathcal{E}}}{\sum_l B_{2,l}^{\mathcal{E}}}, \tag{4.12}$$

we can find the relation between our descriptor and third Ihara coefficient

$$c_3 = -\frac{\mu^{\mathcal{E}}(2)}{3}\sum_l B_{2,l}^{\mathcal{E}}. \tag{4.13}$$

Furthermore, the relation to the number of triangles in a graph, links $\mu^{\mathcal{E}}(2)$ descriptor with a *clustering coefficient* invariant (A.3).

It is also interesting to note that coefficient c_5, carrying information about pentagons in G, is related to $G_2^{\mathcal{E}}$ and, correspondingly, to fourth row of $B^{\mathcal{E}}$ matrix. The $G_2^{\mathcal{E}}$ graph is built based on closed walks of length 5, but without backtracking constraint enforced for prime cycles reflected by Ihara coefficients. Therefore, it encodes not only the number of pentagons but also the number of triangles with a single edge attached to one of the vertices.

The number of edges in $G_k^{\mathcal{V}}$ is equal to the number of shortest paths of length k in the graph G. Based on definitions of descriptors $wi(G)$ and $ef(G)$, after applying Equation 4.8, we can express Wiener Index and efficiency of G as weighted sums of vertex B-matrix elements.

$$wi(G) = \frac{1}{2}\sum_k\sum_l klB_{k,l}^{\mathcal{V}} \tag{4.14}$$

$$ef(G) = \frac{1}{n(n-1)}\sum_k\sum_l \frac{l}{k}B_{k,l}^{\mathcal{V}} \tag{4.15}$$

The weights depending on B-matrix row (k) and column (l) affect contribution of non-zero elements of $B^{\mathcal{V}}$ to descriptor value. For Wiener Index, the higher values of k and l yield the greater impact. In turn, the efficiency is high for graphs with quickly growing k-shell sizes, that is for non-zero elements indicated by higher column numbers and lower row numbers. As the product of Wiener Indices for two graphs is equivalent to the shortest-path kernel with linear kernel on shortest paths distances [33], Equation 4.14 can be used to express shortest-path kernel as a product of weighted sums over elements of $B^{\mathcal{V}}$.

The k-th power of adjacency matrix A_G^k allows for counting the number of walks of length k in a graph G (see Theorem 1.4.13). Let us consider the ordered set of (u, v) elements from subsequent adjacency matrix powers, that is $\{A_G^1(u,v), A_G^2(u,v), \ldots\}$. The smallest value of k for which $A_G^k(u,v) \neq 0$ determines the length of the shortest path between u and v. This can be expressed as follows

$$d_G(u,v) = floor_k A_G^k(u,v). \tag{4.16}$$

Therefore, to generate adjacency matrix of graph $G_k^{\mathcal{V}}$ ($A_{G_k^{\mathcal{V}}}$) using adjacency matrix A_G we need to compare sum of first $k-1$ powers of A_G with A_G^k. This sum is frequently referred to

as adjacency matrix of the $(k-1)$-th power of the graph ($A_{G^{k-1}}$). Using boolean operators of negation and conjunction it can be expressed by formula

$$A_{G_k^\vee} = \neg \sum_{i=1}^{k-1} A_G^i \wedge A_G^k = \neg A_{G^{k-1}} \wedge A_G^k, \tag{4.17}$$

where non-zero matrix elements are treated as *true* and zero ones as *false*. Moreover, in the work by Bai et al. [21] it was also shown that

$$d_G(u,v) = floor_k P_G^k(u,v) = floor_k \sum_{i=1}^{n} (1-\lambda_i)^k \phi_i(u)\phi_i(v), \tag{4.18}$$

and

$$h_t(u,v) = exp(-t) \sum_{k=0}^{n^2} P_G^k(u,v) \frac{t^k}{k!}, \tag{4.19}$$

where h_t is the heat kernel of graph G (A.15) and P_k is the modified adjacency matix with a weight $1/\sqrt{deg(u)deg(v)}$ put to edge $\{u,v\}$. The heat kernel of a graph can be expressed as a sum of time-parametrized Poisson distributions over walk-lengths. These equations provide the link between heat kernel descriptors related to walks and distance k-graphs invariants based on shortest paths.

4.3 Experiments

Herein, we describe experiments performed to test pattern vectors derived from distance k-graphs. First, we use artifical datasets and investigate stability of graph embeddings under controlled structural errors. To this end the relationships between graph edit distance and Euclidean distance among new feature vectors are presented. Afterwards, unsupervised learning on synthetic dataset is studied. Next, we demonstrate experiments on supervised graph learning using two real-world datasets: graph representations of satellite photos obtained from *Google Earth* and mutagenicity dataset from IAM benchmark database [132].

The new graph descriptors are compared with graph characteristics generated on the basis of heat kernel [166], that is heat kernel content invariant and heat kernel content coefficients invariant (A.15). We also use control feature vector D_{con} that contains seven well-known scalar graph descriptors such as *efficiency* (A.1) and average values and standard deviations of *degree*, *clustering coefficient* (A.3) and *betweenness* (A.8). The results of graph embeddings are evaluated using clustering validation indices (Appendix C): *C index*, *Davies-Bouldin index* and *Rand index* for which we use 5-nearest neighbor algorithm to establish agreements in cluster assignments [83].

4.3.1 Controlled structural errors

We used connected Erdős-Rényi graph with 220 vertices, 440 edges and diameter 7 as a seed graph for performing random edit operations including edges addition (*ea*), edges deletion (*ed*) and vertices deletion (*vd*). A selected edit operation was performed d times to obtain sample graph at distance d from the seed graph. For a given d we computed 100 samples, whereby $1 \leqslant d \leqslant 120$ (*ea*, *ed* operations) and $1 \leqslant d \leqslant 60$ (*vd* operation). Additionally, for each modified

graph the pattern vectors based on its distance k-graphs were generated. Then, we computed Euclidean distances between those vectors and the seed graph feature vector. The dependencies of Euclidean distance on graph edit distance for two selected graph descriptors $D^{\mathcal{V}}_{long}$ and $D^{\mathcal{V}}_{ef}$ are presented in Figure 4.9. The average values and standard deviations of Euclidean distances are reported.

For all tested descriptors the relationship between Euclidean distance and graph edit distance is approximately linear. Both for $D^{\mathcal{V}}_{long}$ and $D^{\mathcal{V}}_{ef}$ the slope of *ed* line is greater than *ea* line. The same is observed for variance which is greater for delete operation. By removing edges we can obtain non-connected graph for which many pairs (u, v) exist so that $d_G(u, v) = \infty$. This strongly affects structural properties of $\{G^{\mathcal{V}}_k\}$ set, as none of $G^{\mathcal{V}}_k$ graphs is connected. From the perspective of distance k-graphs invariants, vertex/edge deletion is more structure-disturbing operation than edge insertion. This is also visible on relative deviation (RSTD) diagrams (see Figure 4.9c,f). The stability of $D^{\mathcal{V}}_{long}$ descriptor is better than one for D_{ef} pattern vector, which yielded approximately four times greater RSTDs.

4.3.2 Artificial graphs

In order to generate first dataset we selected four seed graphs of size 100: *bb-seed* is 3-regular graph with 150 edges, *me-seed* is irregular 2D mesh with 200 edges, *rm-seed* is 2D lattice with 180 edges and *rr-seed* is random connected Erdős-Rényi graph with 143 edges. For each seed graph we performed 100 series of random edit operations that can be described by three parameters (*ne*, *spread*, *type*). Parameter *ne* denotes a number of edit operations performed, *spread* is a maximal random number added to *ne* in order to increase group variance and *type* describes type of edit operations performed (edge addition, deletions or both). Clusters *bb*, *me* and *rr* were generated using parameters (40, 20, *both*) while *rm* with (20, 30, *deletion*). In this manner we obtained groups that cannot be easily separated by graph density. The sample graphs from the dataset are depicted in Figure 4.10. Except for *rr* sample, the planar embeddings appear to be structurally similar. The differences can be captured more accurately with the use of vertex B-matrices.

The descriptors based on B-matrices were computed and mapped onto 2D and 3D space using two unsupervised dimensionality reduction algorithms PCA (Principal Component Analysis) and LPMIP (Locality-Preserved Maximum Information Projection) [156]. The latter one controls between-locality and within-locality simultaneously. The tradeoff between local and global structure of the data is adjusted using parameter α. In this manner we can mix properties of manifold learning methods such as LLE (Locally Linear Embedding) or LPP (Locality Preserving Projection) with a general-variance methods as PCA. The discrimination ability of introduced descriptors is evaluated with the use of clustering validation indices computed in low-dimensional spaces. The most interesting results are reported in Table 4.1. Low values of *C index* (range $[0; 1]$) and *Davies-Bouldin index* (range $[0; \infty]$) indicate good clustering, whereby *Davies-Bouldin index* prefers clusters of spherical shape. *Rand index* ranges from 0 to 1, where 1 means perfect clustering. The three parameters of LPMIP algorithm were adjusted manually to ($nearest\_neighbors = 20$, $\sigma = 20$, $\alpha = 0.1$), see Table 4.1. This means that we set the weight 0.1 to between locality and 0.9 to within locality.

The best results (both in 2D and 3D space) were obtained for long feature vectors derived from edge B-matrix (see Table 4.1, rows 2 and 10). Figure 4.11 shows the embedding of graphs in 3D space using $D^{\mathcal{E}}_{long}(1, 4, 1, 25) + \mu^{\mathcal{E}}, \mu^{\mathcal{V}}, \sigma^{\mathcal{E}}, \sigma^{\mathcal{V}}$ for $1 \leq k \leq 4$ and its associated distance matrix. With the use of edge B-matrices we can generate robust graph characteristics, that

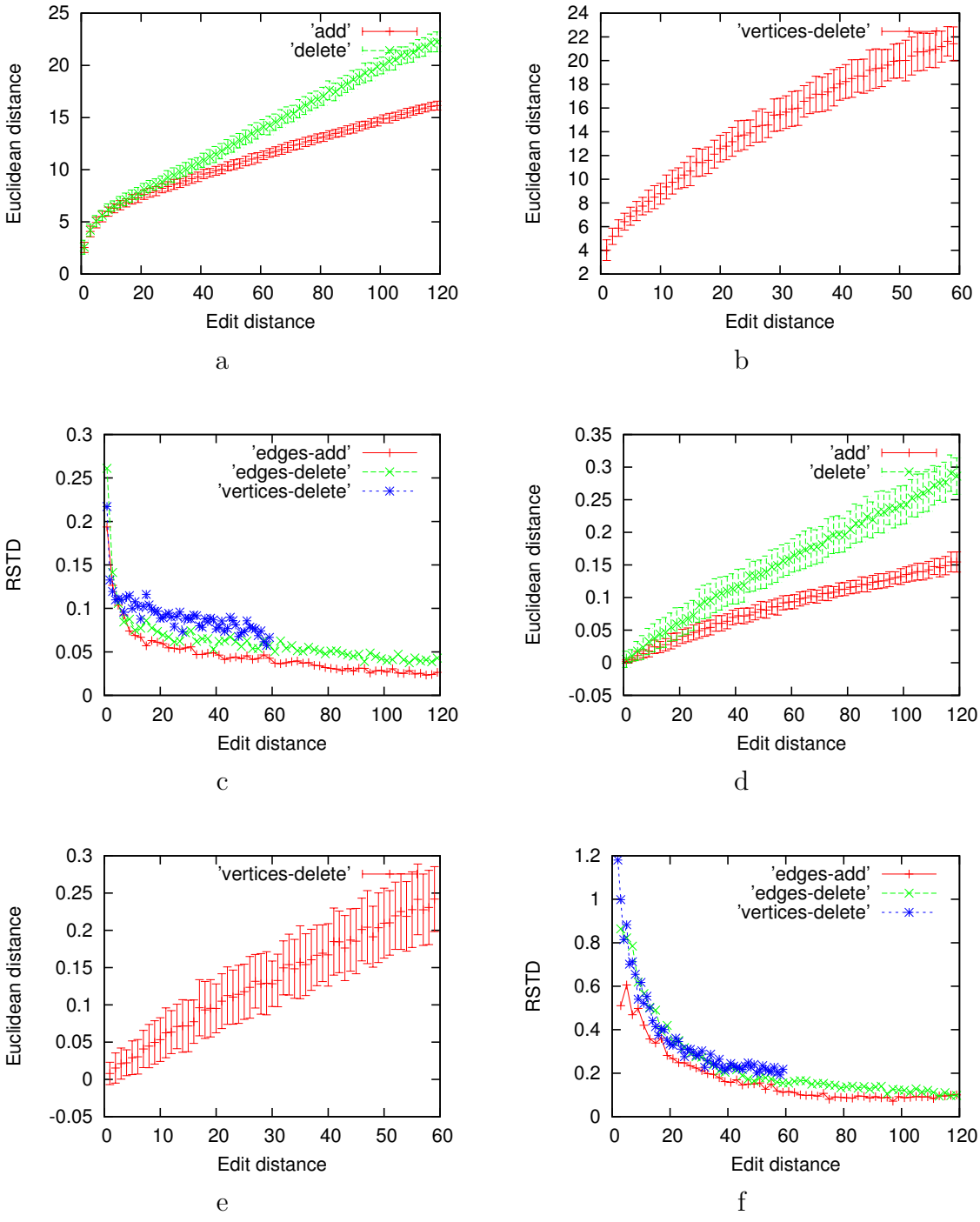


Figure 4.9: Euclidean distance (average value and standard deviation) vs. number of edges/vertices added/deleted for selected graph feature vectors. Charts a, b, c describe $D^{\mathcal{V}}_{long}(1, 7, 1, 50)$ pattern vector with logarithmic scaling: a. edges addition/deletion, b. vertices deletion, c. relative standard deviation for edit operations *ea*, *ed* and *vd*. Charts d, e, f concern $D^{\mathcal{V}}_{ef}(k)$, $1 \leqslant k \leqslant 15$ descriptor: d. edges addition/deletion, e. vertices deletion, f. relative standard deviation for all operations.

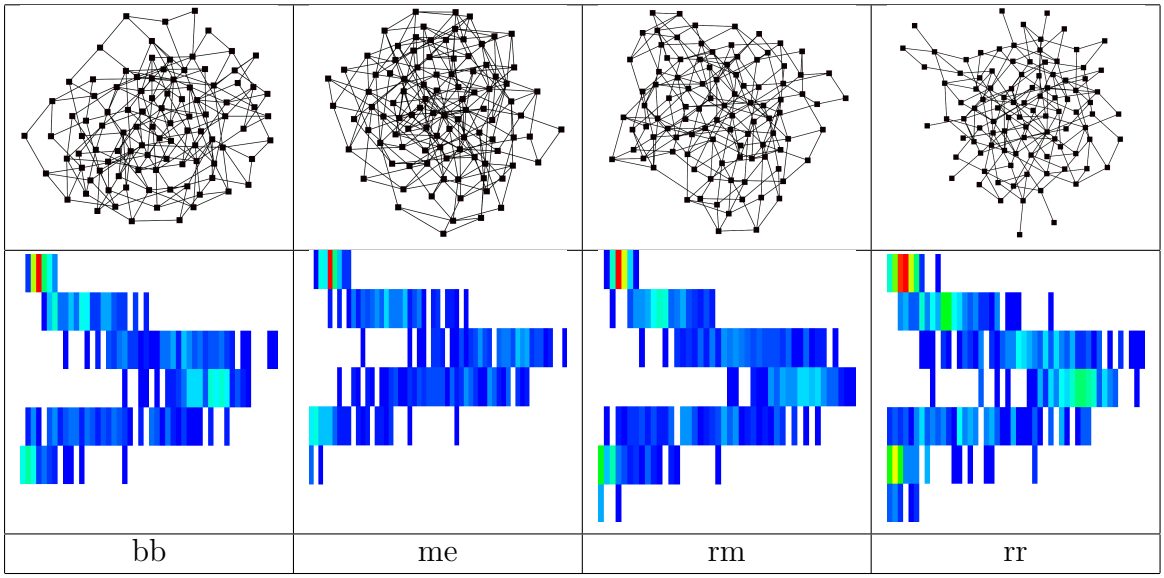


Figure 4.10: Visualization of sample graphs from artificial dataset using planar embeddings and vertex B-matrices.

	Vector	Dim	Dim Red Method	2D	3D
1	$D^{\mathcal{E}}_{long}(1,4,1,25)$	100	PCA	0.07, 1.00, 0.93	0.05, 1.13, 0.96
2	$D^{\mathcal{E}}_{long}(1,4,1,25)$	100	LPMIP(20, 20, 0.1)	**0.04, 0.69, 0.98**	**0.03, 0.74, 0.99**
3	$D^{\mathcal{V}}_{long}(1,4,1,20)$	80	PCA	0.13, 1.62, 0.89	0.14, 1.85, 0.94
4	$D^{\mathcal{V}}_{long}(1,4,1,20)$	80	LPMIP(15, 20, 0.1)	0.08, 2.11, 0.93	0.08, 2.13, 0.93
5	$D^{\mathcal{V}}_{ent}$, $1 \leq k \leq 10$	10	PCA	0.14, 1.28, 0.79	0.15, 1.45, 0.80
6	$D^{\mathcal{E}}_{avgd}$, $1 \leq k \leq 20$	20	LPMIP(20, 20, 0.1)	0.12, 1.67, 0.81	0.15, 1.96, 0.81
7	$\mu^{\mathcal{E}}, \mu^{\mathcal{V}}, \sigma^{\mathcal{E}}, \sigma^{\mathcal{V}}$, $1 \leq k \leq 4$	16	PCA	**0.07, 1.16, 0.97**	**0.11, 1.50, 0.97**
8	$D^{\mathcal{V}}_{rstd}$, $1 \leq k \leq 10$	10	LPMIP(20, 20, 0.1)	0.09, 1.2, 0.86	0.15, 1.69, 0.87
9	$D^{\mathcal{E}}_{rstd}$, $1 \leq k \leq 20$	20	LPMIP(20, 20, 0.1)	0.09, 1.37, 0.81	0.16, 1.92, 0.84
10	vectors from row 2 and 7 together	116	LPMIP(20, 20, 0.1)	**0.01, 0.47, 1.0**	**0.04, 0.54, 1.0**
11	D_{hkc}, $1 \leq t \leq 10$	10	PCA	0.05, 0.73, 0.89	0.05, 0.73, 0.98
12	D_{hkc}, $1 \leq t \leq 20$	20	PCA	0.05, 0.86, 0.86	0.05, 0.86, 0.97
13	D_{hkcc}, $1 \leq m \leq 10$	10	PCA	0.07, 1.46, 0.79	0.07, 1.47, 0.78
14	D_{hkcc}, $1 \leq m \leq 20$	20	PCA	0.09, 1.16, 0.80	0.09, 1.17, 0.80
15	D_{con}	7	PCA	0.11, 2.01, 0.79	0.11, 2.14, 0.86

Table 4.1: Low-dimensional embeddings of descriptors derived from graph B-matrices evaluated using indices: *C index*, *Davies-Bouldin index* and *Rand index* (Appendix C). The order of indices preserved in triples presented in the table

can perform better than ones derived from a heat kernel. Moreover, the cost of computing such descriptors is lower ($\mathcal{O}(n^2)$ vs. $\mathcal{O}(n^3)$). The $B^{\mathcal{E}}$ stores more discriminative information than $B^{\mathcal{V}}$. Long vectors perform better than aggregated statistics (e.g. $D^{\mathcal{E}}_{rstd}$), nevertheless the latter are still comparable with descriptor D_{hkcc}, that requires $\mathcal{O}(n^3)$ steps (see Table 4.1, rows 7 and 8).

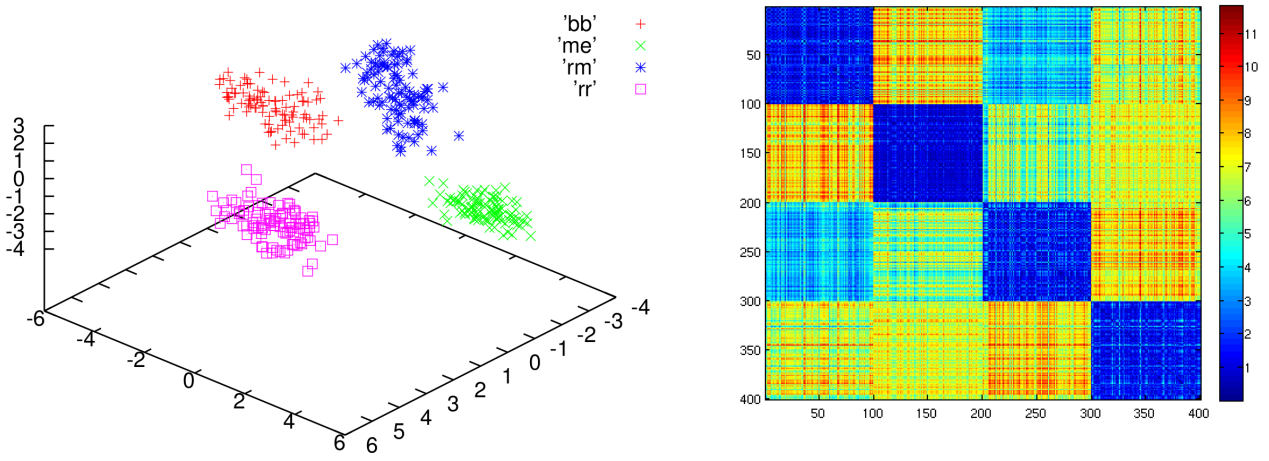


Figure 4.11: 3D embedding of combined feature vector build from graph B-matrices (see Table 4.1, row 10) and its associated distance matrix.

4.3.3 Metabolic networks

Structure of metabolic networks change with the evolution of organisms forming a good basis for assessing inter-species similarity and recognition of taxonomic groups (see Section 3.3.1 for a more detailed description of metabolic networks). In this experiment we study clusterization of organisms on the basis of B-matrices of their metabolic networks . Only structural information is used in comparison, we skipped chemical compound labels attached to graph vertices.

The dataset comes from CCNR networks database [3] and contains substrate-reaction metabolic networks of selected 43 organisms. We filtered out networks with a highest and lowest number of vertices obtaining set of 23 species: 4 Eukaryotes, 14 Bacteria and 5 Archaea (see Table 4.2). Next, the GCC of each graph was extracted yielding final dataset with network sizes varying from 658 to 1268 and average size 968. In the unsupervised learning experiment we compared results of extracting two principal components (PCA) from a matrix of features constructed using different graph descriptors. Particularly, we investigated graph descriptors derived from B-matrices. In addition to PCA we employed *Locally Linear Embedding* (LLE) dimensionality reduction algorithm which preserves neighborhood structure of high-dimensional data. The results are reported in Table 4.3 with the best ones visualized also on a plane (see Figure 4.12). Next, the distance matrices generated on the basis of two best low-dimensional representations (see Table 4.3, row 1 and row 3) were used to generate phylogenetic trees (neighbor joining hierarchical clustering) with a help of PHYLIP application [4].

The obtained phylogenetic trees are depicted in Figure 4.13. In our experiment we computed descriptors derived from vertex B-matrices only, because for bipartite substrate-reaction metabolic networks even rows of edge B-matrices are empty, so $B^{\mathcal{E}}$ does not bring additional information. The parameters of descriptors were selected arbitrarily, so that the number of B-matrix rows included (k_{max}) equals maximum average diameter of metabolic network. In order to decrease number of columns in B-matrix, the size of column bin was set to 10.

Estimating taxonomic groups from the structure of metabolic networks is a typical unsupervised task, in which non-parametrized learning methods are most convenient to use. In case of PCA projection, the best results were obtained for D_{long} descriptor aggregating 18 left-most columns of B-matrix. The descriptors $D^{\mathcal{V}}_{ent}$ and $\mu^{\mathcal{V}}, \sigma^{\mathcal{V}}$ also perform well in terms of general

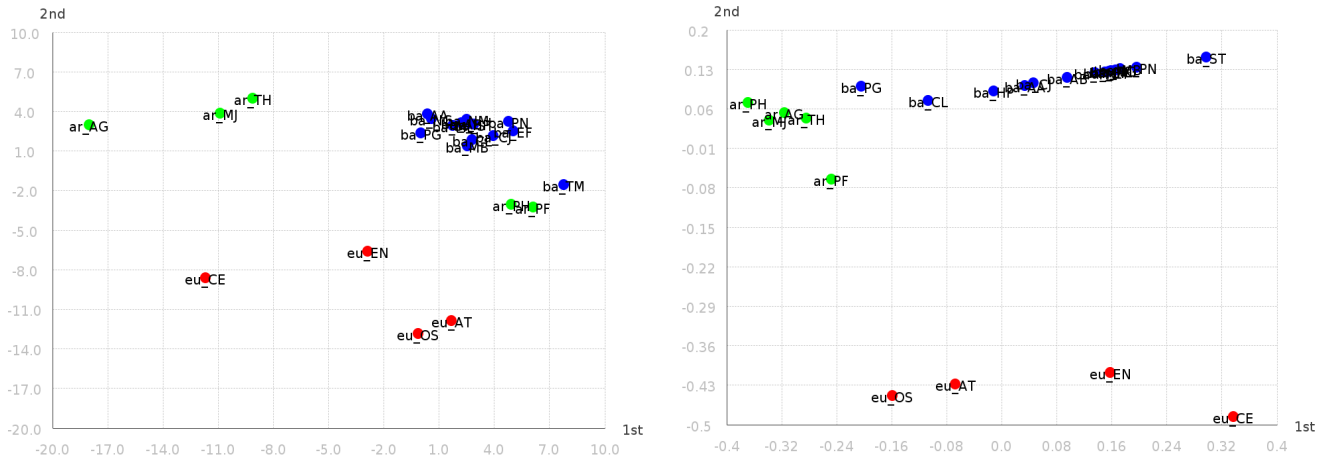
Organism	Label	Domain	Vertices	Edges
Arabidopsis thaliana	eu_AT	Eukaryotes	689	1579
Caenorhabditis elegans	eu_CE	Eukaryotes	1173	2842
Emericella nidulans	eu_EN	Eukaryotes	911	2158
Oryza sativa	eu_OS	Eukaryotes	658	1498
Aquifex aeolicus	ba_AA	Bacteria	1052	2497
Actinobacillus actinomycetemcomitans	ba_AB	Bacteria	991	2335
Campylobacter jejuni	ba_CJ	Bacteria	939	2221
Chlorobium tepidum	ba_CL	Bacteria	913	2138
Enterococcus faecalis	ba_EF	Bacteria	1000	2431
Helicobacter pylori	ba_HP	Bacteria	940	2281
Mycobacterium bovis	ba_MB	Bacteria	1038	2441
Mycobacterium leprae	ba_ML	Bacteria	1050	2489
Neisseria gonorrhoeae	ba_NG	Bacteria	1042	2519
Neisseria meningitidis	ba_NM	Bacteria	970	2348
Porphyromonas gingivalis	ba_PG	Bacteria	1001	2322
Streptococcus pneumoniae	ba_PN	Bacteria	1069	2591
Streptococcus pyogenes	ba_ST	Bacteria	1043	2525
Thermotoga maritima	ba_TM	Bacteria	821	1957
Archaeoglobus fulgidus	ar_AG	Archaea	1268	2980
Methanococcus jannaschii	ar_MJ	Archaea	1082	2561
Pyrococcus furiosus	ar_PF	Archaea	746	1749
Pyrococcus horikoshii	ar_PH	Archaea	764	1779
Methanobacterium thermoautotrophicum	ar_TH	Archaea	1112	2682

Table 4.2: Metabolic networks from CCNR database [92].

	Vector	Dim	Dim Red Method	2D
1	$D^{\mathcal{V}}_{long}(1,14,1,18)$	252	PCA	**0.14**, **1.9**, **0.87**
2	$D^{\mathcal{V}}_{rstd}$, $1 \leq k \leq 14$	14	PCA	0.19, 3.6, 0.70
3	$D^{\mathcal{V}}_{rstd}$, $1 \leq k \leq 14$	14	LLE(8)	**0.16**, **0.8**, **0.96**
4	$\mu^{\mathcal{V}}, \sigma^{\mathcal{V}}$, $1 \leq k \leq 14$	28	PCA	0.13, 3.29 , 0.74
5	$\mu^{\mathcal{V}}, \sigma^{\mathcal{V}}$, $1 \leq k \leq 14$	28	LLE(7)	0.13, 1.52 , 0.70
6	$D^{\mathcal{V}}_{ent}$, $1 \leq k \leq 14$	14	PCA	**0.12**, **2.31**, **0.78**
7	$D^{\mathcal{V}}_{ent}$, $1 \leq k \leq 14$	14	LLE(11)	0.13, 1.60, 0.61
8	$D^{\mathcal{V}}_{ef}$, $1 \leq k \leq 14$	14	PCA	0.17, 1.13, 0.83
9	$D^{\mathcal{V}}_{ef}$, $1 \leq k \leq 14$	14	LLE(8)	0.2, 1.07, 0.91
9	D_{hkc}, $1 \leq t \leq 30$	30	PCA	0.18, 5.26, 0.61
10	D_{hkc}, $1 \leq t \leq 30$	30	LLE(11)	0.33 , 3.66, 0.61
11	D_{hkcc}, $1 \leq m \leq 30$	30	PCA	0.26, 3.74, 0.52
12	D_{hkcc}, $1 \leq m \leq 30$	30	LLE(9)	0.43, 4.19, 0.61
13	D_{con}	7	PCA	0.20, 3.03, 0.65
14	D_{con}	7	LLE(8)	0.25, 2.91, 0.61

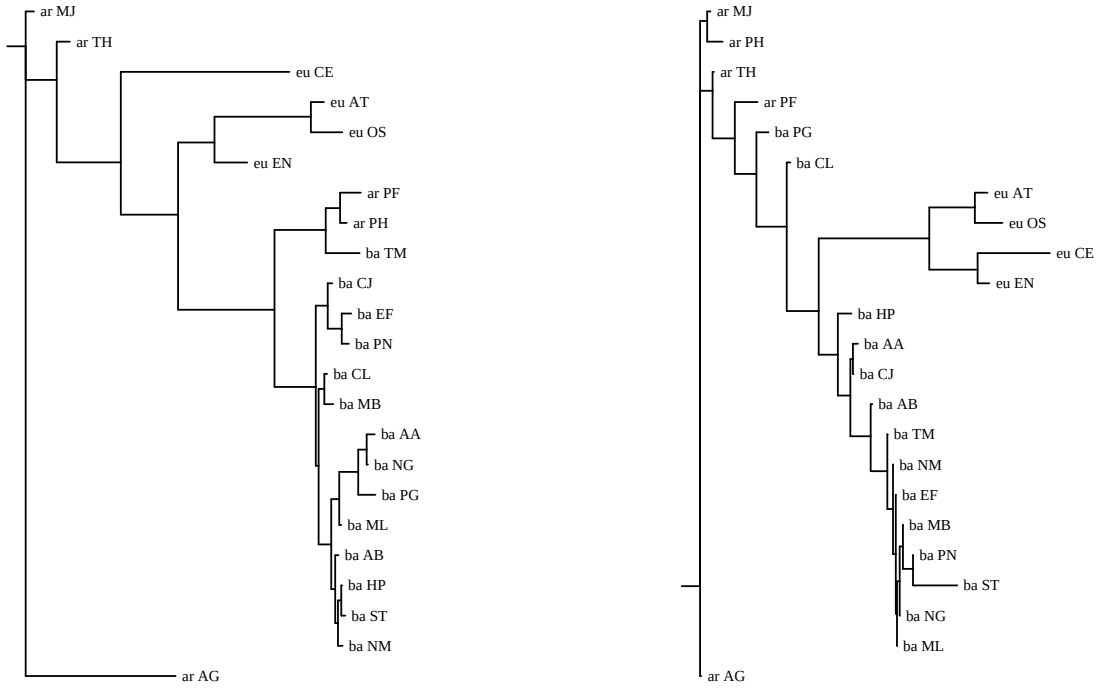
Table 4.3: 2D embeddings of descriptors derived from B-matrices of metabolic networks, evaluated using indices: *C index*, *Davies-Bouldin index* and *Rand index* (the order of indices preserved in triples presented in the table). In the LLE algorithm we used fixed regularization parameter $r = 1.0^{-5}$ and nearest neighbors parameter (displayed in parentheses) adjusted to obtain best separation of clusters.

variance. The clustering validation indices computed for embeddings based on heat kernel matrix and control feature vector indicate much worse separability of clusters. Even relatively low-dimensional $D^{\mathcal{V}}_{ent}$ (14D) is more informative than elaborate D_{hkc} (30D) descriptor. For neighborhood-preserving LLE projection the reported number of neighbors used to calculate point reconstruction weights is one for which we obtained highest values of clustering validation indices. After applying LLE the cluster separation for $D^{\mathcal{V}}_{rstd}$ and D_{hkcc} descriptor was improved



a b

Figure 4.12: Visualization of 2D embeddings of pattern vectors representing metabolic networks, a. PCA (see Table 4.3, row 1), b. LLE (see Table 4.3, row 3)



a b

Figure 4.13: Phylogenetic trees generated from distance matrices of the set 2D pattern vectors representing metabolic networks of 23 organisms: a. $D^{\mathcal{V}}_{long}$ descriptor + PCA (see Table 4.3, row 1) b. $D^{\mathcal{V}}_{rstd}$ + LLE (see Table 4.3, row 3)

with the former descriptor yielding best overall result. The control 7D descriptor constructed from typical scalar descriptors gave poor results as well as spectral descriptors. Indeed, as

indicated in the work [17], such a measures are frequently not discriminative enough to capture similarity of metabolic networks. In turn, B-matrices appear to be a good choice for extraction of relevant features of such graphs.

Taxonomic groups emerging from distance matrices based on D_{long} and D_{rstd} descriptors correctly reflect high level domain taxons, whereby separation for D_{rstd} + LLE is more clear (see Figure 4.13b). Nevertheless, the variance-maximizing approach presented in Figure 4.13a brings several interesting, low-level groupings. For instance, TM (*Thermotoga maritim*a) bacteria located in a three-member subcluster consisting also of hypher-thermophilic archaea PF (*Pyrococcus furiosus*) and PH (*Pyrococcus horikoshii*) is known as one of deepest lineages in *Eubacteria* kingdom, with 24% of genes similar to *Archaea* genes. The optimal growth temperatures for those organisms are as follows: 80°C (TM), 98°C (PH) and 100°C (PF). The species AT and OS representing kingdom *Plantae* are close to each other on both trees.

4.3.4 Satellite photos

In this section we describe experiments on classification of graphs generated from satellite photos. With the help of *Google Earth* application we obtained three groups of photos (size 1412×940) representing fragments of cities. Each group contains 90 samples from approximately the same area taken from altitudes 980m to 1.08km. In order to simulate occlusions and clutter, prior to photo exporting, we performed rotations and translations of a scene. Three examples from this dataset are depicted in Figure 4.14. The photos were transformed to graphs using Harris corner detector [88] and Delaunay triangulation. To discard graph size as a main discrimination factor we decided to select 100 most distinct corners. As all photos contained at least 100 corners, 270 graphs of size 100 were generated.

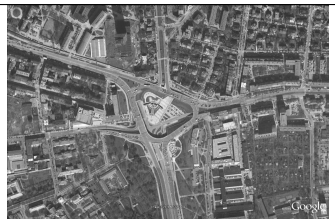
Photo			
Instances	90	90	90
Label	BO	MO	OK

Figure 4.14: Three samples from satelite photo dataset obtained using *Google Earth* application.

We used this dataset to study the performance of the nearest centroid classifier for feature vectors based on distance k-graphs. Our results were compared with ones received for graph characteristics derived from a heat kernel matrix [166]. We randomly selected 15 graphs of each class to construct the test set and use the remaining graphs as the training set (75 training samples, 15 testing samples). The operation was performed 200 times for each type of graph feature vector. The average and maximum classification accuracy is reported together with its standard deviation. Additionally, in order to get a meaningful low-dimensional representation of patterns and increase recognition rate we applied dimensionality reduction methods PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis) and MMC (Maximum Margin Criterion) [106]. The projection vectors were computed using training data and then applied to obtain lower-dimensional representations of vectors from the testing set, like in

Fisherfaces or Eigenfaces technique [28]. The results of classification are depicted in Table 4.4. The target dimensionalities were selected experimentally to obtain highest accuracies.

Vector	Dim	Dim Red	Target Dim	Accuracy		
				Avg	Max	StdDev
$D^{\mathcal{E}}_{avgd}$, $1 \le k \le 20$	35	PCA	10	0.72	0.87	0.06
$D^{\mathcal{E}}_{rstd}$, $1 \le k \le 15$		PCA	15	0.73	0.89	0.06
		LDA	2	**0.77**	**0.91**	0.06
		MMC	2	0.62	0.82	0.08
		MMC	7	0.67	0.82	0.07
$D^{\mathcal{V}}_{ef}$, $1 \le k \le 10$	20	LDA	2	0.65	0.87	0.07
$D^{\mathcal{V}}_{Est}$, $1 \le k \le 10$		MMC	3	0.61	0.76	0.07
$D^{\mathcal{E}}_{ef}$, $1 \le k \le 20$	40	PCA	15	0.65	0.84	0.07
$D^{\mathcal{E}}_{Est}$, $1 \le k \le 20$		MMC	3	0.67	0.84	0.06
$D^{\mathcal{V}}_{long}(1,8,1,30)$	240	MMC	3	**0.75**	**0.93**	0.06
$D^{\mathcal{E}}_{long}(1,20,1,100)$	2000	MMC	100	**0.78**	**0.93**	0.06
D_{hkc} and	30	PCA	10	0.65	0.80	0.07
D_{hkcc}, $1 \le t \le 15$		PCA	15	0.66	0.80	0.08
		LDA	2	0.64	0.80	0.06
		MMC	2	0.67	0.84	0.06
D_{con}	7	PCA	5	0.60	0.71	0.08
		LDA	2	0.69	0.84	0.07
		MMC	2	0.68	0.80	0.06

Table 4.4: Comparison of average and maximal recognition rates for graph descriptors derived from distance k-graphs and heat kernel matrix.

The results obtained for descriptors derived from distance k-graphs are considerably better than classification rates for D_{hkc}, D_{hkcc} and D_{con}. Even unsupervised, global-variance-oriented dimensionality reduction with PCA brings better accuracy than D_{hkc} with LDA or MMC. The highest average accuracies were achieved for $D^{\mathcal{E}}_{long}(1,20,1,100)$ and for aggregated $D^{\mathcal{E}}_{avgd}$ and $D^{\mathcal{E}}_{rstd}$ descriptors. Dimensionality of $D^{\star}_{long}$ descriptors exceeds the number of samples, therefore, in this case we applied MMC, which does not suffer from small sample size problem and, additionally, is less vulnerable to overfittig (for high input dimensionality LDA has poor generalization capability). The accuracies for $D^{\mathcal{V}}_{Est} + D^{\mathcal{V}}_{ef}$ vector are closer to ones achieved for heat kernel based descriptors. Single feature per distance k-graph does not provide enough discrimination information.

4.3.5 Mutagenicity dataset

Mutagenicity dataset from the IAM database [132] contains labeled graphs representing molecular compounds. The graphs are grouped in two classes *mutagen* and *non-mutagen*. The atoms are represented by vertices (symbols *C*, *O*, *H*, *N*, *Cl* as vertex labels) while bonds are modeled by edges. Predicting mutagenicity of a molecule is a task of significant importance as mutagens are often responsible for tumor genesis. The sizes of graphs range from 6 to 417 with average value 30.3. Besides, the dataset is divided into training (1500 instances), validation (500 instances) and testing (2337 instances) sets.

In this experiment we investigate if purely structural representation of molecules (attributes skipped) and distance k-graphs invariants can form a good basis for classifying mutagens. We compare our results with reference classification rate 71.5% achieved for this dataset by Riesen

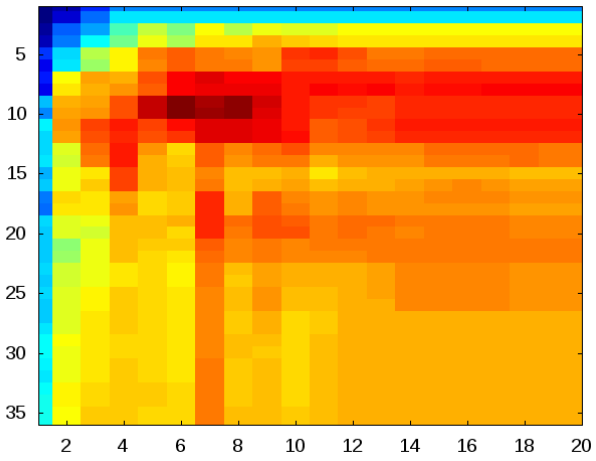


Figure 4.15: Classification rates on mutagenicity validation set obtained using 1NN classifier and $D^{\mathcal{E}}_{long}(1, k_{max}, 1, l_{max})$ feature vector. Vertical axis shows values of k_{max}, while horizontal axis values of l_{max}. The best accuracies were obtained for $k_{max} = 9, l_{max} = 6$ and $k_{max} = 10, l_{max} = 6$.

and Bunke [132] using k-nearest neighbor classifier and graph edit distance dissimilarity (vertex and edge attributes used). We focus on $D^{\mathcal{E}}_{long}$ descriptor which yielded best accuracy in experiments described in section 4.3.4. First, the pattern vector $D^{\mathcal{E}}_{long}(1, 80, 1, 20)$ was computed for each instance in training and testing sets. The initial values of parameters $k_{min}, k_{max}, l_{min}, l_{max}$ were adjusted so that entire edge B-matrix of each graph is encoded by a long vector. To assess gains in classification rates we refer to *Zero* classifier which chooses the class with the greatest number of members in training data. For mutagenicity dataset it yields 55.33%. We obtained accuracy 68.72% for initial pattern vector $D^{\mathcal{E}}_{long}(1, 80, 1, 20)$.

Vector/FS	Dim	Classifier	Accuracy	Remarks
		Zero	55.33%	Classifer selects majority class in training data
$D^{\mathcal{E}}_{long}(1, 80, 1, 20)$	1600	1NN	68.72%	
$D^{\mathcal{E}}_{long}(1, 11, 1, 6)$	66	1NN	**71.12**%	Descriptor parameters values determined on validation set
$D^{\mathcal{E}}_{long}$/Gain Ratio [85]	162	1NN	70.39%	Feature ranking on training set
$D^{\mathcal{E}}_{long}$/Principal Components	83	1NN	68.93%	Training set
$D^{\mathcal{E}}_{long}$/Genetic search [85]		1NN	69.58%	Wrapper subset selection on validation set, population size 20
$D_{hkc} + D_{hkcc}$	40	1NN	61.15%	$1 \leq t \leq 20$

Table 4.5: Classification results for mutagenicity dataset.

Next, in order to reduce input dimensionality and improve classification rate on the testing set, we performed feature selection by adjusting values of $D^{\mathcal{E}}_{long}$ parameters with the use of the validation set. The $D^{\mathcal{E}}_{long}$ descriptor is characterized by four parameters $k_{min}, k_{max}, l_{min}, l_{max}$

which extract rectangular submatrix from respective B-matrix. Exhaustive search of 4D parameter space with constraints $1 \leqslant k_{min} < k_{max} \leqslant 80$, $1 \leqslant l_{min} < l_{max} \leqslant 20$ and application of 1NN classifier on validation dataset can be used to determine descriptor parameters. Nevertheless, such a process is time consuming as 1NN classification on 500-element validation set is performed 160000 times. Therefore we propose to fix values of two parameters $k_{min} = 1$, $l_{min} = 1$ and optimize k_{max} and l_{max}. With increasing value of k_{max} we are moving from local to global features. Growing l_{max} allows for including information about edges with a higher centrality. The classification rates obtained on validation set for $1 \leqslant k_{max} \leqslant 36$ and $1 \leqslant l_{max} \leqslant 20$ are depicted in Figure 4.15 in a form of heat map. We achieved best accuracy 75.4% for $k_{max} = 9, l_{max} = 6$ and $k_{max} = 10, l_{max} = 6$, therefore we use those parameters and their neighboring values $k_{max} \pm 1$, $l_{max} = \pm 1$ for classification on the test set. The best result obtained here is 71.12% ($k_{max} = 11$, $l_{max} = 6$, see Table 4.5) which is 0.38% worse than the accuracy reported by Riesen and Bunke. Nevertheless, this classification rate was acquired with less computational cost and using only structural information. For a comparison, in Table 4.5 we also report results obtained using different feature selection methods such as Gain Ratio ranker [124], principal components and wrapper subset selection with 1NN classifer on validation set and genetic search [85]. Our method of determining descriptor parameters yielded best accuracy.

4.4 Discussion

In this chapter we have demonstrated how to extract features from graphs using distance k-graphs invariants. Particularly, we investigated B-matrix representation constructed on the basis of distance k-graphs degree distributions. Experiments have been conducted to study the stability of proposed graph embeddings and test their discrimination abilities. The graph feature vectors derived from B-matrices were successfully employed to distinguish graphs with subtle structural differences. Classification performed on satellite photo and mutagenicity datasets proved that new pattern vectors can outperform descriptors based on heat kernel matrix. Furthermore, for sparse graphs B-matrices can be constructed with $\mathcal{O}(n^2)$ steps what makes this approach more computationally feasible than the eigendecomposition-based algorithms. Exploiting GPU implementations of all-pair shortest-paths algorithm can bring additional performance boost to the presented approach. We have adjusted parameters of long pattern vector based on edge B-matrix and showed that non-optimal feature subset selection can improve classification rate on mutagenicity dataset. Using our approach we obtained better results than for other ranking and wrapper feature selection methods.

The main limitation of presented graph comparison methodology is that the size of B-matrix representation varies with graph diameter and size. If we encounter high variance of those variables in an analyzed dataset, the selection of right descriptor parameters becomes troublesome. The question whether to perform logarithmic scaling or row normalization also arises and the answer is not straightforward. Aggregating scalar distance k-graphs invariants such as efficiency is an easier approach, however as we demonstrated in an experimental study, it may not bring satisfactory results due to lower information content.

In the future we aim to test our graph comparison method on weighted graph datasets. We also plan to investigate different vertex-vertex metrics such as commute time [123] and adopt vertex-vertex similarity measures like f-communicability [65]. Furthermore, subsequent tests on benchmark graph data will be performed.

Chapter 5

Graph Investigator application

5.1 Related work

Network analysis software developed so far provide a variety of methods to investigate structured datasets. However, their practical use is often limited to the graphs of specific type, e.g. social networks. What follows, the set of available descriptors is domain-biased. Also, group analysis using unsupervised learning techniques is most frequently unavailable. Existing applications share many features, for instance majority of them possess a visualization module or structure edition module. Besides, the set of available graph descriptors is often modest, containing basic measures and lacking more specific ones.

The example of large network analysis tool is *Pajek* [27], a Windows application capable of visualizing graphs and performing statistical analysis of their topology. Despite its advantages such as stability and maturity, platform dependence and closed code decrease flexibility of this application. The *NetworkX* Python package [5] provides a productive framework for the study of networks but requires programming skills, moreover the set of implemented graph measures is modest. The tools intended for statistical social networks analysis and modelling are represented by *StOCNET* [90] or *Visone* [36] applications. *StOCNET*, an open source Windows program, does not provide analytic tools from spectral graph theory and its application to networks other than social is limited. *Visone* is a closed source Java application with strong visualization capabilities. It allows for computation of vertex metrics but more advanced features such as forming feature vectors and embedding graph into pattern space are not available. From the perspective of spectral graph theory, investigation of networks structure can be carried out using *Spectral net* [68] program. Unfortunately, this .NET application seems to be no longer maintained. Commercial graph exploration software is exemplified by *NetMiner* [6] or *orgnet* [7] frameworks.

Our motivation for starting work on *Graph Investigator* was to create an application focused on computing rich set of graph descriptors and capable of network feature generation for comparing real-world networks with some model network types. As pointed out in Section 3.4.1 the specific motivation was to provide tool for comparison of tumor blood vascular networks obtained from simulation and *in vivo* experiments. To this end, we developed application module intended for analysis of groups of graphs using graph embedding and statistical pattern recognition algorithms. To be a self-contained framework *Gin* comprises other modules such as visualization, B-matrix analysis, statistical analysis, etc. The new method of graph comparison based on distance k-graphs, presented in Chapter 4 is also incorporated into our software framework.

5.2 Program description

Graph Investigator is a program written in Java, available under (`http://home.agh.edu.pl/czech/gin`) as *Java Web Start*. We decided to use *Java* programming language to provide platform independence and take advantage of robust graph libraries as *JUNG* [8] and *Prefuse* [9]. The former package provides number of algorithmic tools while the latter is an advanced visualization framework. The application was designed to be extended easily, hence new functionality e.g. new graph descriptors can be added without much effort. Provided that Java Runtime Environment is installed, *Graph Investigator* can run on many different platforms as Linux, Solaris or Windows.

The functionality of *Graph Investigator* is described below. First we present graph data formats that can be used with our application. Then the basic analytic tools are described. Next we report available visualization modules and finally we show how to perform analysis of sample datasets.

5.2.1 Input/Output

Graph Investigator works with real-world or artificial networks. The latter ones can be generated using the following models: Barabási-Albert [23], Eppstein power law [61], Erdős-Rényi [62], Kleinberg small-world [101], Watts small-world, clique model, path model, star model and balanced tree model. Artificial networks are frequently used for comparison with real-world data. Such analysis can reveal rules governing creation of real-world networks. *Graph Investigator* accepts the following graph input formats: simple list of edges (*.el), Graph ML (*.gxml), Amira skeleton (*.am) [10], Pajek (*.net), edgelist (*.edge), adjacency matrix (*.mtrx), Graph eXchange Language (*.gxl) and binary Amalfi graph format [67]. A detailed description of these file formats is available in application documentation. A user can also save graph with the use of part of the above-mentioned formats and perform command line format conversion for large sets of input data. Uploaded networks are presented in the table (see Figure 5.1) with tooltips providing general information such as number of edges and number of vertices.

5.2.2 Graph descriptors

The analysis of network structure can be performed with the help of 98 parametrized descriptors which are described in Appendix A and Appendix B. *Graph Investigator* provides two modes of invariants computation: graph descriptors and vertex/edge descriptors. Separate measures can be aggregated into one feature vector and used in this form in further analysis (e.g. using *Principal Component Analysis*). Computation of vertex-attribute-based statistical moments such as mean, standard deviation, skewness and kurtosis is also possible. *Graph Investigator* enables to draw histograms of edge/vertex/path descriptors. As shown in Figure 5.1, graph descriptors can be presented in a table while edge/vertex descriptors in a edge/vertex tooltip on a graph picture. Values of vertex/edge descriptors can be also shown in a graph visualization window using predefined colormap (see Figure 5.1). Spectral analysis module enables to calculate algebraic decompositions (Eigendecomposition, SVD) of several graph matrices such as adjacency matrix, Laplacian or normalized Laplacian. It also provides comparison of spectral density functions for selected graphs. Additionally, application allows for computation of descriptors based on heat kernel matrix [166]. Graph measures can be saved in CSV format

(*Comma Separated Values*).

5.2.3 Visualization

Graph Investigator provides three modes of graph visualization: with force simulation, using B-matrices of a graph [57] and using spectral embedding [78, 150]. The parameters of force simulation can be adjusted to obtain different graph views. The vertex or edge descriptors are visualized as tooltips or colors on graph drawing. The results of graph visualization can be saved in the following graphic file formats: pdf, svg, jpg, png.

5.2.4 Graph analysis

Graph Investigator handles two types of datasets: single graph and groups of graphs. As described in Section 5.2.2 graph invariants can be computed for each dataset type. In case of grouped graphs, unsupervised and supervised learning based on user-defined feature vectors can be performed. The available dimensionality reduction methods include *Principal Component Analysis* (PCA), *Locally Linear Embedding* (LLE) [136], *Isomaps* [147], *Kernel PCA* [114], *Locality-Preserved Maximum Information Projection* (LPMIP) [156], *Linear Discriminant Analysis* (LDA), *Maximum Margin Criterion* (MMC) [106]. Three indices are used for validation of low-dimensional embeddings: *C index*, *Davies-Bouldin index* and *Rand index*. The computed feature vectors can be saved as attribute-relation file (*.arff) and used further with the Weka machine learning software. For importing to Matlab or Octave text file matrix output is provided, and Gnuplot script output for results visualization.

5.2.5 Other features

Additionally, we provide useful feature for visualization of clustering. The set of graphs representing certain real-world objects can be embedded into lower-dimensional space, where datapoints are visualized as mini-photos (photos representing objects are associated to data as described in application documentation). Moreover, the application allows for: extracting greatest connected component, extracting all connected components, testing graph descriptors using random edit operations, adjacency matrix matching via *Singular Value Decomposition*, adjacency matrix visualization and B-matrix comparison for single graph and groups of graphs.

5.3 Sample use cases

5.3.1 Normal brain vascular network

The complex network discussed in this section represents brain cerebral micro-vascularization, derived from 3D images acquired by confocal microscopy [70]. The moisac of 3D images can be transformed into 3D skeleton using algorithm introduced in [70]. The skeleton, described by the set of points and curves, is read by *Graph Investigator* and converted into graph with 2206 vertices and 2983 edges. Spatial visualization of this brain vascular network is depicted in Figure 5.2a. For the sake of simplicity, in further analysis, we take into account its greatest connected component (2131 vertices and 2846 edges). The Figure 5.2b presents visualization of this component using force simulation (*Graph Investiagtor*).

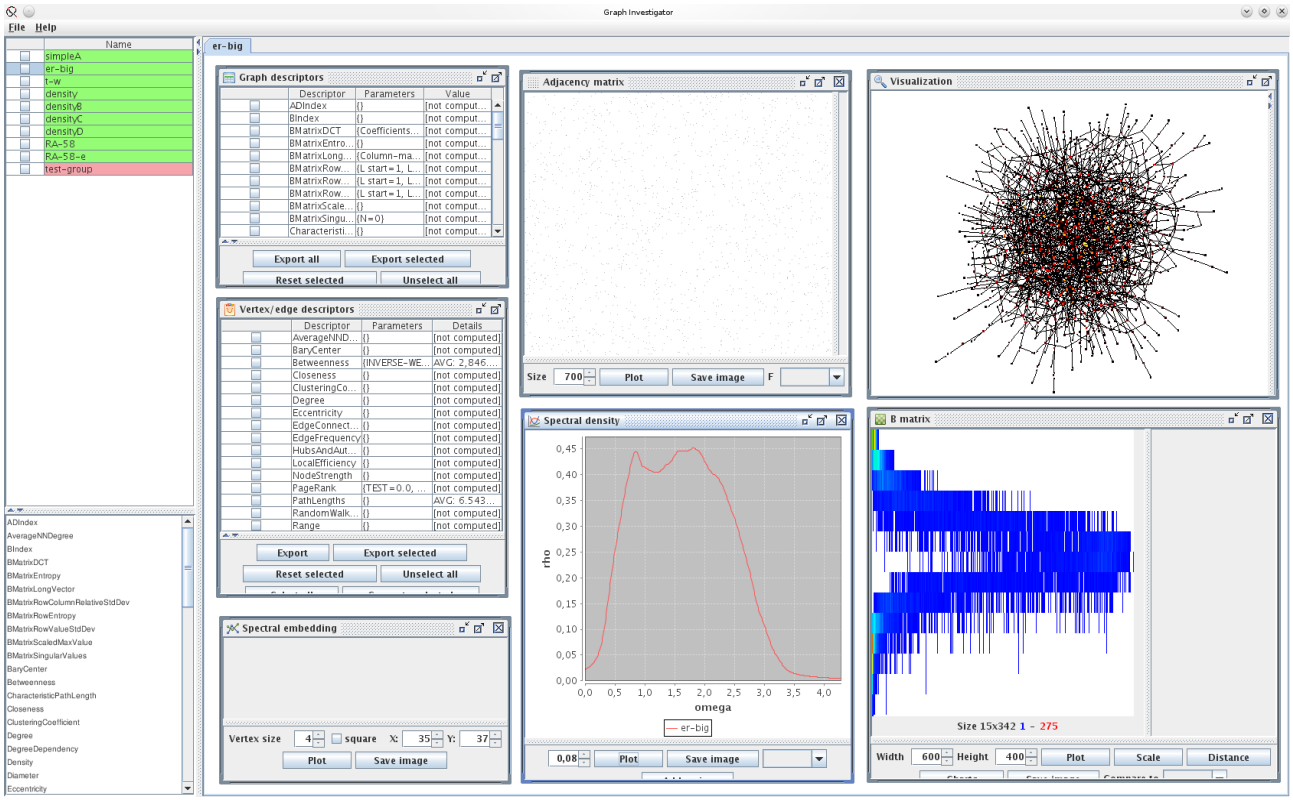


Figure 5.1: The main window of *Graph Investigator*. On the left - graph panel with a table containing single-graph and multi-graph datasets. The main panel on the right shows internal frames that allow for graph analysis and visualization.

a b

Figure 5.2: Visualization of brain vascular network using *Amira* (a) and *Graph Investigator* with force simulation (b). Picture (a) reflects real locations of vessels in space, whereas (b) emphasizes topological features with better visible endings (forks) and long paths.

In Figure 5.3 the histogram of vertex degrees in vascular network is depicted. The network has nearly uniform distribution of vertex degrees, nevertheless vertices with number of neighbors greater than 3 also occur. The vertices of valence 3 dominate, which is a typical feature of river networks or tree branches, that possess three-way junctions (forks) most likely. A relatively high number of degree 1 vertices appears due to boundary cut of images from confocal microscopy (see Figure 5.2a).

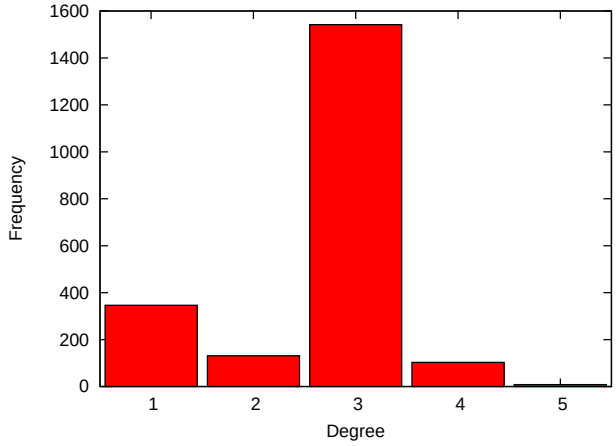
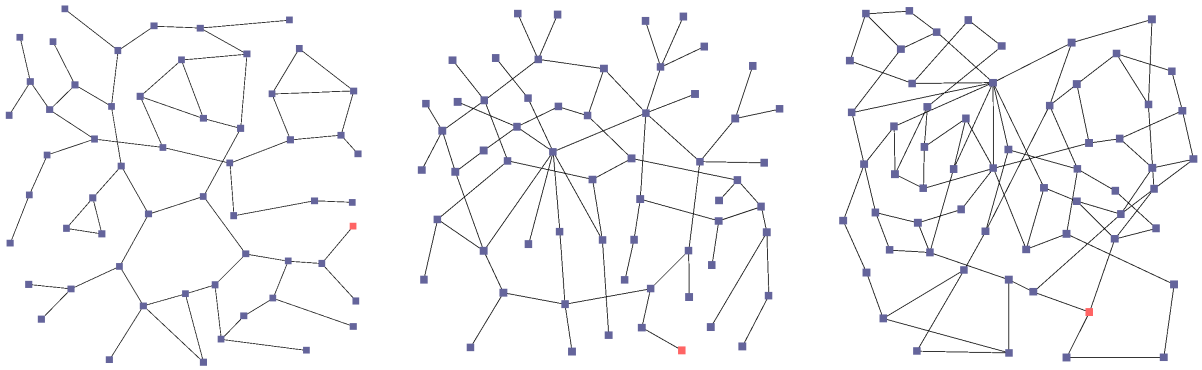


Figure 5.3: Brain vascular network degree histogram

We compare the brain vascular network with two artificial networks of similar density and average degree using subset of descriptors available in *Graph Investigator*. Our goal is to determine if its structure is random, not revealing any patterns or to uncover the features that make vascular network different from resembling artificial networks. First random graph was generated using Erdős-Rényi model [62] in which an edge is established between each pair of vertices with equal probability r, being the parameter of the model. The density of Erdős-

Rényi graph depends on r. An example of such a graph for $r = 0.01$ and 60 vertices is depicted in Figure 5.4b. Graphs obtained using Erdős-Rényi model can be unconnected. In this case we add extra edges to get connected network. The second model allows to generate irregular graphs [67], in which average valence is bounded by parameter k. The graph is build on the basis of fixed valence random graph by moving some part of edges to new locations. Here we decided to replace 10% of edges selected with uniform probability. After indicating edges to move, network vertices are permuted so that each vertex receives new index i. For a given edge new endpoint i is chosen using probability function $\alpha \exp(-\beta i)$, where α and β depend on graph size and are normalized as described in [67]. An example of irregular bounded valence graph is depicted in Figure 5.4c ($k = 3$).



a b c

Figure 5.4: Examples of graphs from dataset used to test proximity of artificial and vascular networks: a. fragment of brain vascular network (57 vertices), b. Erdős-Rényi graph ($r = 0.01$, 60 vertices) b. irregular bounded valence graph ($k = 3$, 60 vertices).

A subnetwork consisting of 1090 vertices and 1438 edges was extracted from brain vascular network by random selection of node and following *breadth-first* traversal with depth limited to 15. Then, we created two networks of similar size and density using Erdős-Rényi ($r = 0.001$) and irregular bounded valence models ($k = 3$). The set of graph descriptors computed for three networks is presented in Table 5.1.

We assumed that the size and average vertex degree are similar for all three networks. As presented in Table 5.1 some descriptors differ slightly while the rest capture features that separate analyzed graphs. For instance normalized information of vertex degrees (see Appendix A) is similar for all graphs. Moreover, relative differences in Randić connectivity index are also minor. It reflects that general connectivity and complexity understood as number of cliques, branches etc., is similar for three considered networks (vascular network has slightly smaller structural complexity). On the other hand, values of B index, normalized Wiener index, M_1 Zagreb index, normalized Platt index and standard deviation of degree distinguish vascular network from artificial ones. These descriptors are similar for artificial networks whereas vascular network is characterized by values approximately two times larger or smaller. This indicates larger branching of vascular network, which seems to possess more forks and bifurcations. The distribution of vertex degrees in vascular network is less wide than in case of artificial graphs. As far as distribution of clustering coefficients and ${}^{m}M_1$ Zagreb index are considered, vascular network is more similar to Erdős-Rényi graph than to irregular bounded valence graph. Yet, in terms of two statistical metrics computed on the basis of vertices betweenness centrality (standard deviation and kurtosis) vascular network is closer to irregular bounded valence graph.

Table 5.1: Descriptors computed for three networks: vascular, Erdős-Rényi ($r = 0.001$) and irregular bounded valence ($k = 3$)

Descriptor	Network		
	Vascular	Erdős-Rényi	Bounded valence
Number of vertices	1090	1028	997
Number of edges	1438	1582	1500
Diameter	30	15	18
Avg of Degree	2.639	3.078	3.01
StdDev of Degree	0.827	1.552	1.612
Kurtosis of Degree	0.411	0.260	40.699
Skewness of Degree	-0.958	0.711	5.588
M1 Zagreb index	8334	12212	11616
MM1 Zagreb index	293.04	266.96	163.23
B index	0.195	0.497	0.426
Avg of Clustering Coefficient	0.222	0.153	0.033
StdDev of Clustering Coefficient	0.381	0.358	0.177
Kurtosis of Clustering Coefficient	0.218	1.796	25.537
Skewness of Clustering Coefficient	1.406	1.943	5.224
Normalized Wiener index	0.013	0.006	0.007
Avg of Betweenness Centrality	7163.1	2846.6	3205.6
StdDev of Betweenness Centrality	10068.7	3069.0	6321.8
Kurtosis of Betweenness Centrality	22.208	3.863	60.535
Skewness of Betweenness Centrality	3.698	1.766	6.816
Avg of RW Betweenness Centrality	0.0243	0.0123	0.0153
StdDev of RW Betweenness Centrality	0.0167	0.0076	0.0128
Estrada Index	3344.0	3834.4	3853.9
Normalized information of vertex degrees	0.0004	0.0005	0.0005
Normalized Platt index	4.2e-6	8.4e-6	8.7e-6
Randics connectivity index	526.3	481.6	477.8
Normalized Gordon-Scantterburry index	6.5e-9	2.3e-8	2.9e-8

To complete the comparison of the networks, we present their B-matrices computed using shortest-path and commute time metric (see Figure 5.5). The B-matrices of three graphs differ significantly. This is particularly visible for shortest-paths-based edge B-matrices, for which number of non-empty rows is determined by a graph diameter. The matrices constructed from commute time reveal less diverse patterns, however they also can be used to distinguish three considered networks.

Our next step is to divide large vascular network into smaller subgraphs (possibly overlapping in part) and try to measure to what extent such a group of graphs is structurally close to clusters of artificial networks. In this experiment we selected two groups of random graphs from the database described in [67]: the set of 20 connected Erdős-Rényi graphs (60 vertices, $r = 0.01$, denoted by ER) and the set of 20 irregular bounded valence graph (mean valence 3, denoted by BV). The group of 19 graphs was generated on the basis of large brain vascular network by random selection of core node and *breadth-first* traversal with limited depth. Only graphs of the size greater than 50 were considered. No more than 30 vertices may overlap between two different instances. This set of vascular network fragments is denoted by FV. In the Figure 5.4, three selected instances (one for each group) are presented. The averages of mean vertex degrees are as follows: FV 2.37, ER 2.32, BV 2.99.

On the basis of computed graph descriptors, we created multi-dimensional feature vectors to analyze structural proximity between groups of networks. Then, the pattern vectors were

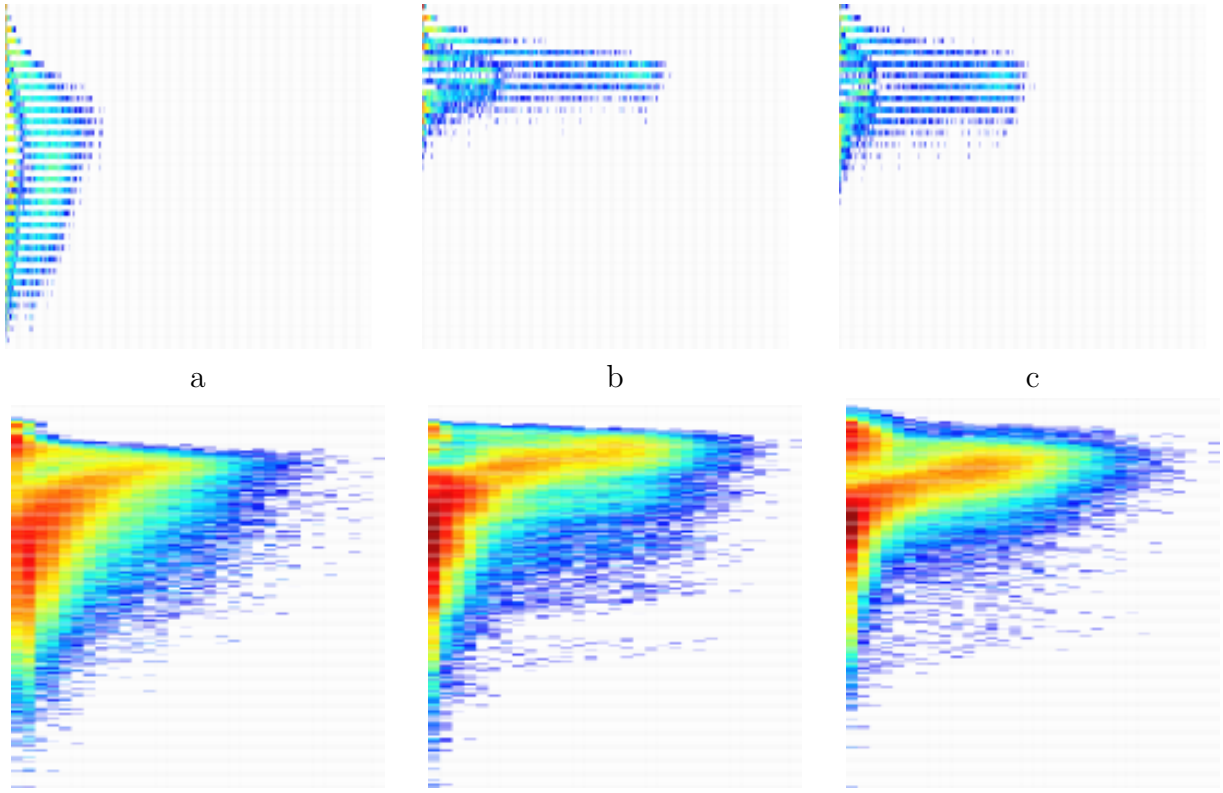


Figure 5.5: B-matrices of three networks computed using shortest-path and commute time metric. First row (a, b, c) depicts edge B-matrices based on shortest-paths, while second row (d, e, f) B-matrices obtained from commute time metric. First column (a, d) represents brain vascular network (1090 vertices, 1438 edges), second column (b, e) describes connected Erdős-Rényi network ($r = 0.001$, 1028 vertices, 1582 edges), third column (c, f) concerns irregular bounded valence network ($k = 3$, 997 vertices, 1500 edges). The logarithmic color scale and common matrix size are used to emphasize proportions of histograms.

embedded into 2D space using *PCA* (*Principal Component Analysis*). We tested a number of combinations of descriptors, each time performing dimension reduction and evaluation of clusters separation using two validation indices *Davies-Bouldin index* and *C index* (in 2D). The best result for 4D feature vector composed of general graph descriptors: standard deviation of degree, standard deviation of clustering coefficient, normalized Platt index and normalized Wiener index is presented in Figure 5.6a. An example of 2D embedding using 2D spectral graph descriptor is depicted in Figure 5.6b. This feature vector consists of mean value and standard deviation of graph Laplacian matrix spectrum. As we pointed out earlier (see Table 5.1), standard deviation of degree separates vascular networks from Erdős-Rényi and irregular bounded valence graphs, while standard deviation of clustering coefficient increases distance between irregular bounded valence network and the rest.

Brain vascular network has specific structure distinguishing it from random graphs created on the basis of artificial models. The distribution of its vertex degrees is nearly uniform, with small variance. We observe relatively high clustering coefficients, branching factors and average centralities of nodes. In case of two betweenness centrality measures, this network has the greatest average values that indicate its efficiency in transporting blood both through shortest

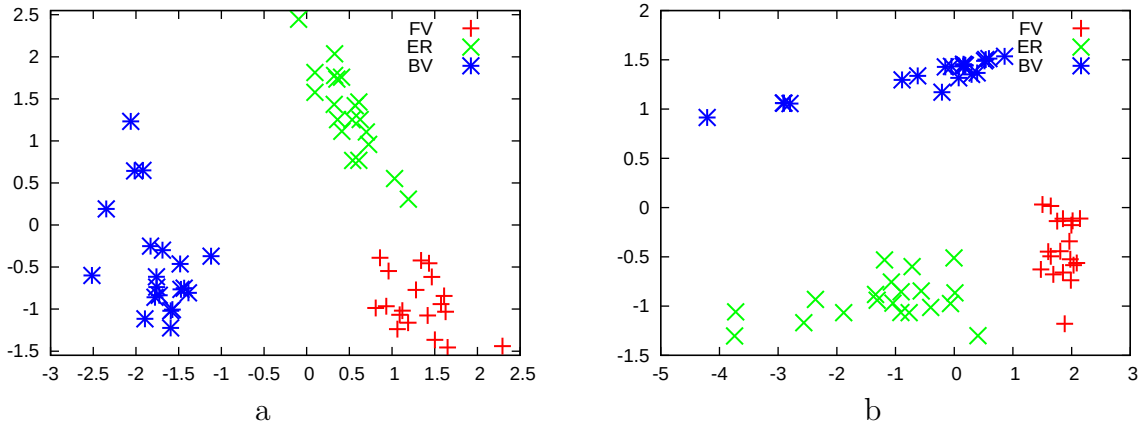


Figure 5.6: Three groups of brain vascular networks embedded into 2D space (PCA) using two feature vectors: a. 4D vector composed of standard deviation of degree, standard deviation of clustering coefficient, normalized Platt index and normalized Wiener index, b. 2D vector formed of mean value and standard deviation of Laplacian matrix eigenvalues.

and random paths. These topological features are correlated with circulatory system function of delivering chemicals to tissues and are apparently the result of evolutionary optimization.

5.3.2 Tracking angiogenesis simulation

The use case presented in this section is based on simulation data obtained from cellular automata model of tumor-induced angiogenesis [148]. We consider four sets of evolving vascular networks obtained for different parameters of the model and present how the properties of these networks change during simulation.

The model that generates described networks consists of two interacting parts: a transportation network (blood vessels) and consuming environment (tissue) [148]. The topology of the network changes in response to distribution of TAFs (Tumour Angiogenesis Factors) epitomized by VEGF (Vascular Endothelial Growth Factor) [42], which are produced by starving tumor cells in order to stimulate vessels growth. In turn, the vascular network delivers oxygen and nutrients to the tissue forming a gradient which affects tumor cells. The tissue is modelled by a mesh of cellular automata while the transportation network by a graph of cellular automata built over the CA mesh. The TAF concentration exceeding certain threshold activates vessels to create sprouts that develop towards tumor tissue. The vessel states evolve from immature to mature. Vessel maturity reflects its ability to transport blood and form new branches.

The model is characterized by a number of parameters governing rules of vascular network evolution. In this work we focus on three types of parameters that control branching: TAF threshold T_c, baseline branching probability P_b and level threshold multipliers that provide a functional relationship between TAF concentration and branching probability. Provided that TAF threshold in certain location is exceeded, the relevant vascular cell can start branching with certain probability P_b. This probability can be adjusted using two pairs of parameters: threshold T_1, multiplier m_1 and threshold T_2, multiplier m_2. If $T_c > T_1$ then P_b is multiplied by m_1 and if $T_c > T_2$ then branching probability is set to $m_2 P_b$. The parameters above allows for increasing branching probabilities in case of high TAF concentrations, so that branching activation function has more than one step (see Figure 5.7).

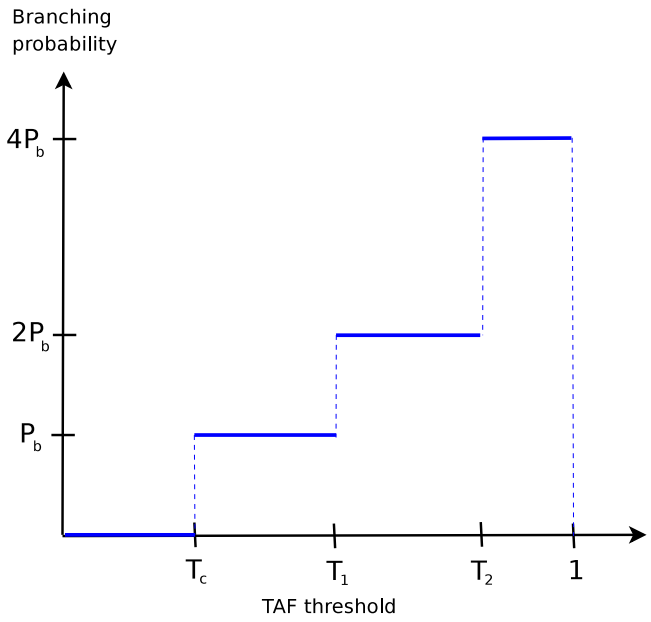


Figure 5.7: Example of branching probability multipliers ($m_1 = 2$, $m_2 = 4$)

In creating angiogenesis models, an important goal is to measure how values of parameters influence growth of vascular network. Graph descriptors are a main tool in analyzing the changing topology of the growing network. We run 4 simulations with different parameters (see Table 5.2). Next, for each simulation step we compute graph descriptors measuring local structural properties of a generated network (mean degree and clustering coefficient). As shown in Figure 5.8a, after 40 steps the sizes of networks start to grow exponentially with different rates influenced by branching probabilities. The fastest growth is observed for set02, characterized by highest branching probability for immature (young) cells. Comparing it to the slowest growth for set04, which has the same T_{cm} value (TAF threshold for mature vessels), similar T_{ci} (TAF threshold for immature vessels) and parallel multipliers for same thresholds it seems that among presented parameters P_{bi} (baseline branching probability) affects growing rate to the greatest extent.

In Figure 5.8 we present graph size dependencies of several metrics. Our aim is to investigate network evolution without explicit time reference and to focus on observing structural differences for networks of the same size. We investigate distributions of two local vertex descriptors: clustering coefficient and degree (see Figure 5.8b, c, d). We observe that clustering coefficient curves for set02 and set04 have similar shape with decrease at the beginning (graph sizes 40-100), increase in the middle (characteristic point at 120 where they separate from the rest) and decrease in the final phase. The decrease of clustering coefficient at the beginning reflects fast path-like growth of vascular networks towards tumor (number of vertices of degree 2 increases, minor branching rate). For degree charts similar separation (set02 and set04 vs. set01 and set03) is observed (characteristic point at 120). It seems that TAF threshold for immature vessels is the discrimination factor here, as its possesses similar value for set02 and set04 (0.4 and 0.5 respectively) and different value for set01 and set03 (0.1). Relatively high values of T_{ci} for set02 and set04 prevent young immature cells (dominating at the beginning of simulation) from forming branches. Nevertheless a few sprouts created from mature cells can develop towards tumor, extending vascular network size (graph sizes 40-100). Therefore after

passing through *maturity* age (approximate graph size 100), the number of *mature* vascular cells increases rapidly, resulting in high branching rate (100 -1000). The growth of sets 01 and 03 is more stable, however for all four sets we observe decrease of clustering coefficient in the final phase. This effect is connected with reaching boundaries of the tissue mesh and periodic boundary conditions.

The analysis illustrates how inspecting single network properties allows for gaining insight into the processes governing the changing topology during network growth. Comparing series of graphs with the use of descriptors allows for revealing network phase transitions and understating how the parameters affect network structure. Also, the results can be used to select the most relevant parameters of the model.

Table 5.2: The parameters of angiogenesis model for 4 simulation runs (only parameters that that vary over sets are presented).

	set 1	set 2	set 3	set 4
TAF threshold [a](*mature*) T_{cm}	0.01	0.01	0.01	0.01
TAF threshold (*immature*) T_{ci}	0.1	0.4	0.1	0.5
Branch probability (*mature*) P_{bm}	0.02	0.01	0.01	0.01
Branch probability (*immature*) P_{bi}	0.02	0.08	0.02	0.02
Level 1 threshold	0.6 multi 1	0.6 multi 2	0.6 multi 2	0.6 multi 2
Level 2 threshold	0.8 multi 1	0.8 multi 4	0.8 multi 6	0.8 multi 6

[a] TAF concentration is a real value between 0 and 1

5.4 Discussion

Graph Investigator provides a variety of network analytic tools ranging from feature generation to visualization. It enables to obtain distinctive insights into network structure by employing numerous descriptors from graph theory. The program was built with a special focus on biological networks as this kind of data grows constantly and its analysis has a wide range of applications. We reported three typical use cases: comparison of brain vascular network with artificial networks and inspecting how parameters of angiogenesis model affect structure of generated vascular networks. We believe that graph-theoretical approach to biological data analysis may bring many benefits, therefore developing software aimed at network exploration is a task of great importance.

Graph Investigator was designed to be flexible and easily extended. Graph descriptors can be added as *plugins*. The use of *Java* programming language provides its portability and makes the development simpler and faster. Currently our application is at beta stage, therefore all suggestions and contributions to its development are welcomed.

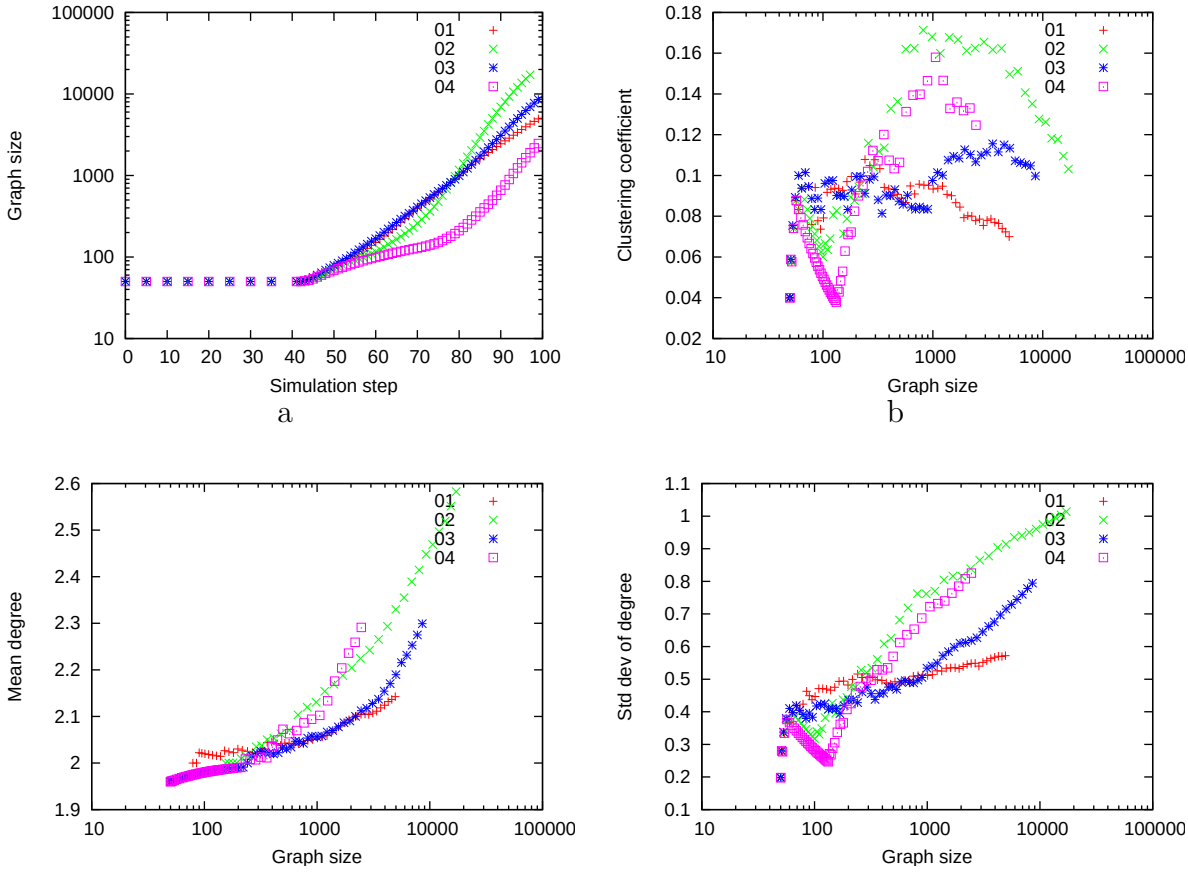


Figure 5.8: The evolution of graph topology measures during angiogenesis simulation a. simulation step dependence of network size (logarithmic Y scale) b. graph size dependence of clustering coefficient (logarithmic X scale) c. graph size dependence of mean degree (logarithmic X scale) d. graph size dependence of standard deviation of degree (logarithmic X scale).

Chapter 6

Efficient graph comparison and visualization using GPU

As it was shown in the previous sections, the practical approach to network analysis requires robust descriptors and efficiency of their computation. The researcher should be able to capture non-trivial graph features and compare or visualize large graphs interactively. Here is where we can benefit from "1 teraflop on desk" provided by massively parallel architecture of modern GPUs. High computational power and low cost of such units allow for increasing the size of analyzed graphs by two orders of magnitude. Nevertheless the algorithms used for computation of graph descriptors and graph visualization should be redesigned to be implemented in GPU programming models like CUDA and applicable to SIMD processor arrays. Such an adaptation includes problem decomposition performed to take advantage of data locality and using data structures suitable for storage of large graphs in the device memory [86].

The motivation of this part of the dissertation is to bring interactive framework for graph comparison, mining and visualization capable of dealing with large graphs and taking advantage of GPU implementations of classical graph algorithms and linear algebra operations. We show that computation of many graph descriptors can be reduced to different variants of the shortest path problem or to a sequence of matrix multiplications. As recently shown, these operations can be executed on GPU with high speed [38, 86]. We briefly present GPU implementation of graph all-pair shortest-paths (APSP) recursive Kleene algorithm [38], breadth-first-search algorithm and describe how they were integrated with existing *Graph Investigator* [58] application via Java Native Interface (JNI). The experiments on selected real-world datasets are reported and conclusions are drawn at the end of the chapter.

6.1 Distance-based graph invariants

In this section we list graph descriptors that can be computed on the basis of information about shortest-paths or using linear algebra operations on graph matrices such as *adjacency matrix* or *Laplace matrix*. These types of computations can be efficiently executed on GPUs, resulting in significant performance boost.

- B-matrices based on shortest-paths and related descriptors (Section 4.2)
- Efficiency (A.1)
- Graph diameter (A.6)

- Radius of a graph (B.1)
- Wiener index (A.2)
- Average path length (A.5)
- Vertex distance (B.2)
- Vertex eccentricity (B.2)
- Vertex closeness (B.2)
- Spectral descriptors (A.15)
- Weighted clustering coefficient (A.4)

Besides, several manifold learning methods such as *Isomaps* [147] rely on distance matrix of k-nearest neighbor graph constructed from data points and can be accelerated using fast implementations of APSP algorithm.

6.2 Short introduction to CUDA

CUDA (Compute Unified Device Architecture) is a parallel computing framework built on Nvidia graphics processing units (GPUs). It provides API interfaces for High Performance Computing (HPC) supporting many programming languages including C, C++, Fortran, Python and Matlab. CUDA forms abstraction layer over GPU computational elements such as memory and processor cores. GPUs devote more transistors to processing than to control flow and caching therefore they are suitable for performing data parallel computations. Today, GPUs like GeForce GTX 590 provide 2488 GFlops single precision peak performance and significant 327.7 GB/s bandwidth.

The scalable programming model of CUDA allows for mapping the same program to devices with a different number of cores. The basic abstraction of CUDA is a *kernel*, which is a function executed in parallel by a certain number of threads. The fine-grained CUDA threads are organized in blocks, which can have 1D, 2D or 3D structure. The *thread block* may contain at most 1024 threads. This constraint stems from the fact that all threads in a block are executed on a single processor core sharing its limited resources. Thread blocks are organized in a 1D, 2D or 3D grid and are executed independently, in any order, what allows for kernel launches on CUDA devices with a different number of processing units. The number of thread blocks in the grid reflects the size of input data. The threads in a block can communicate via shared memory and also may be synchronized. The memory of a GPU is hierarchically organized, starting from local thread memory visible to a single thread, afterwards shared memory accessible for all threads in a block and finally reaching global memory visible to all threads. The global memory access latency is the greatest one.

At hardware level CUDA device is an array of streaming multiprocessors (SM). The number of processor cores in a single SM depends on GPU architecture and varies from 8 to 48 in the newest devices. The threads from a block are executed concurrently on a multiprocessor,

which implements Single-Instruction Multiple-Data (SIMD)[1] processing schema. The instructions form a stream and are executed in order without branch prediction. The threads are managed in groups called *warps*. Each *warp* contains 32 threads which start execution from the same instruction and countinue processing supervised by the control unit of SM issuing consecutive commands [13]. If the execution paths of all warps match, we obtain best efficiency. Nevertheless, the threads can also branch, what forces *warp* to execute sequential part of single thread and disable others. Due to branch divergence the performance of a computation deteriorates. Taking into account all multiprocessors, hundreds of lightweight threads can be active simultaneously on a GPU.

Heterogenic programming in CUDA requires combining host code (CPU) and device code (GPU). Efficiency of the last one is influenced by many factors and often a single line of code can have great impact on the execution time. In order to maximize global memory throughput, during development of a parallel GPU-enabled application one needs to address memory coalescing issues. Also data locality which allows for the use of L1 cache, plays a significant role in producing maximum performance [12]. The next concern is providing high occupancy of the device, i.e., keeping *warps* busy to hide latencies. Finally, the low-level instruction optimization and using specialized function implementations from CUDA API can bring program speedup [12]. The need to tackle all these performance aspects and usually some additional ones, related to application-specific constraints, makes development of efficient GPU applications not a straightforward task.

6.3 Graph algorithms on GPU

Graph descriptors presented in Section 6.1 can be computed efficiently using GPU realizations of two path problems and basic linear algebra routines. In the next sections we address implementation details of these GPU graph algorithms.

6.3.1 All-Pair Shortest-Paths

Highly optimized implementation of all-pair shortest-paths problem using recursive Kleene (R-Kleene) algorithm was recently described in [38]. It uses parallelized, *in-situ* version of fast matrix multiplication routines on a tropical semiring (e.g. Volkov and Demmel version [154]) and stores whole distance matrix in the device memory. Assuming single precision this limits the size of feasible graphs to 39796 on Nvidia Tesla C2070 with 6GB memory. If a graph is undirected, two-bytes `short` type is used for storing distance matrix. This allows for increasing maximum graph size to 56281 vertices, on the same GPU. The basis of R-Kleene algorithm is the correspondence between matrix multiplication and connecting paths in a graph.

Let us recall classical Floyd-Warshall algorithm for all-pair shortest-paths (APSP) problem (see Function 1). It employs dynamic programming and three nested loops processing pattern to compute distance matrix of a weighted graph. The similar approach is used in matrix-matrix multiplication routines (see Function 2). The difference is that in case of MM, the loops can be placed in any order, while for APSP the order of vertices indicated by the outermost k loop cannot change during execution. This makes MM more eligible for loop interchange optimization. The functions listed below can be rewritten in linear algebra notation using operations on

[1] Nvidia uses separate term Single-Instruction Multiple-Thread (SIMT) to distinguish GPU architecture from aproaches denoted by SIMD acronym used mainly in vector architecture contexts

Function 1 Floyd-Warshall APSP

```
public void floydWarshall(float[][] x) {
    int n = x.length;
    for(int k = 0; k < n; k++) {
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++) {
                x[i][j] = min(x[i][j], x[i][k] + x[k][j]);
            }
        }
    }
}
```

Function 2 Matrix self-multiplication

```
public void matrixSelfMultiplication(float[][] x, float[][] y) {
    int n = x.length;
    for(int k = 0; k < n; k++) {
       for(int i = 0; i < n; i++) {
           for(int j = 0; j < n; j++) {
               y[i][j] = y[i][j] + x[i][k] * x[k][j];
           }
       }
    }
}
```

a semiring $(\mathbb{R}^+, \oplus, \otimes, \mathbf{0}, \mathbf{1})$. The modified versions of algorithms are presented as Algorithm 4 (semiring $R_1 = (\mathbb{R}^+, \min, +, \infty, 0)$) and Algorithm 5 (semiring $R_2 = (\mathbb{R}^+, +, \cdot, 0, 1)$). For each value of variable k, the outer product $X(:,k) \otimes X(k,:)$ is computed and the output matrix is updated. In case of Floyd-Warshall algorithm, X denotes working distance matrix storing cost of going from vertex i to vertex j. Initially it is filled with edge weights and the special value ∞ for such entries $X_{i,j}$ that edge $\{i,j\}$ does not exist in the graph. The semiring R_1 is called *tropical semiring*. It possesses an important property called *idempotence*, which means that $x \oplus x = x \equiv \min(x,x) = x$, for all $x \in \mathbb{R}^+$. Therefore, contrary to MM, the APSP can be computed in place.

Algorithm 4 `Floyd-Warshall APSP in algebraic notation`

Input: X {modified adjacency matrix}
Output: X^* {matrix closure}
1: **for** $k = 0$ to $k - 1$ **do**
2: $\quad X = X \oplus X(:,k) \otimes X(k,:)$ {$(\mathbb{R}^+, \min, +, \infty, 0)$ semiring}
3: **end for**
4: $X^* = X$
5: **return** X

As presented in [60] and [38] the type of computations shown in Listing 4 can be carried out recursively using R-Kleene algorithm. The pseudo-code of this *divide and conquer* algorithm is provided in Listing 6. First, the square matrix X is divided into four blocks A, B, C, D. If $n_1 = \lfloor n/2 \rfloor$ and $n_2 = \lceil n/2 \rceil$, then submatrix A has the size $n_1 \times n_1$, submatrix B - size $n_1 \times n_2$, C is submatrix of size $n_2 \times n_1$ and submatrix D is of size $n_2 \times n_2$. Such a partitioning

Algorithm 5 `Matrix self-multiplication in algebraic notation`

Input: X {input matrix}
Output: $Y = X \cdot X$ {matrix self-multiple}
1: **for** $k = 0$ to $k - 1$ **do**
2: $\quad Y = Y \oplus X(:, k) \otimes X(k, :)$ $\{(\mathbb{R}^+, +, \cdot, 0, 1)$ semiring$\}$
3: **end for**
4: **return** Y

reflects division of vertices into two disjoint sets S_1 and S_2 (Figure 6.1). The sets have similar cardinality with difference 1 existing for odd n. Initially, the block A contains information about edges between vertices from set S_1, while the block D encodes connectivity within set S_2. The blocks B and C complete the image by storing costs of intermediate edges joining vertices from set S_1 and S_2. In the next lines of Algorithm 6, two-recursive calls for S_1 and S_2-induced subgraphs are made together with matrix-matrix multiplications on tropical semiring which update path lenghts by taking into account intermediate edges. The more detailed explanation of operations performed in lines 5 – 12 is presented in the example below.

Algorithm 6 `RKleene`

Input: X {initialized with edge weights}
Output: X {at the end - distance matrix}
1: {using tropical semiring $(\mathbb{R}^+, \min, +, \infty, 0)$}
2: {multiplications done in place}
3: $X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$; {divide matrix into blocks}
4: {vertices in two disjoint sets S_1 and S_2}
5: `RKleene`(A);
6: B=$A \otimes B$;
7: C=$C \otimes A$;
8: D=$D \oplus C \otimes B$;
9: `RKleene`(D);
10: B=$B \otimes D$;
11: C=$D \otimes C$;
12: A=$A \oplus B \otimes C$;

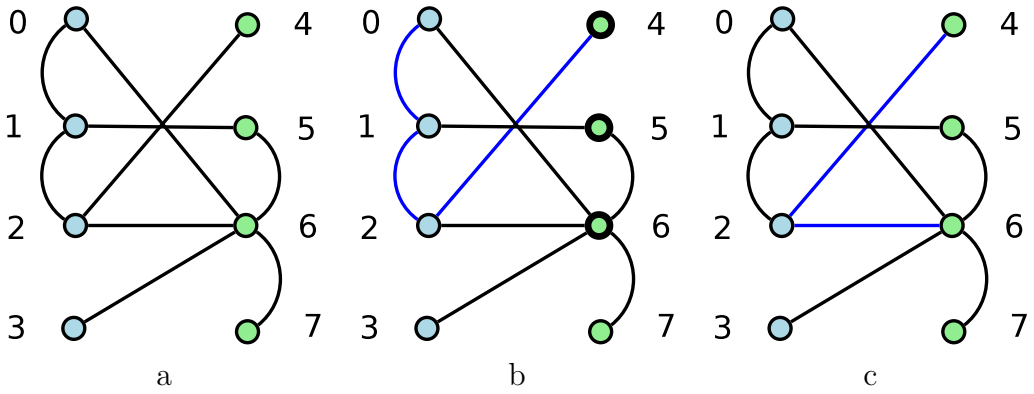


Figure 6.1: Sample graph for demonstration of R-Kleene algorithm. The vertices are divided into two groups: $S_1 = \{0, 1, 2, 3\}$ and $S_2 = \{4, 5, 6, 7\}$.

Example

1. The initial distance matrix X for the graph depicted in Figure 6.1 is presented in Equation 6.1. The matrix is partitioned into four blocks as shown in Listing 6.

$$X = \left[\begin{array}{cccc|cccc} 0 & 1 & \infty & \infty & \infty & \infty & 1 & \infty \\ 1 & 0 & 1 & \infty & \infty & 1 & \infty & \infty \\ \infty & 1 & 0 & \infty & 1 & \infty & 1 & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & 1 & \infty \\ -- & -- & -- & -- & -- & -- & -- & -- \\ \infty & \infty & 1 & \infty & 0 & \infty & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & 0 & 1 & \infty \\ 1 & \infty & 1 & 1 & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & \infty & \infty & \infty & 1 & 0 \end{array}\right], \tag{6.1}$$

2. Let us assume that after the first recursive call for block A (Listing 6, line 5) we obtained distance matrix for a subgraph induced by the set S_1 (see Equation 6.2).

$$A = \left[\begin{array}{cccc} 0 & 1 & 2 & \infty \\ 1 & 0 & 1 & \infty \\ 2 & 1 & 0 & \infty \\ \infty & \infty & \infty & 0 \end{array}\right], \tag{6.2}$$

3. The block B is presented in Equation 6.3.

$$B = \left[\begin{array}{cccc} \infty & \infty & 1 & \infty \\ \infty & 1 & \infty & \infty \\ 1 & \infty & 1 & \infty \\ \infty & \infty & 1 & \infty \end{array}\right], \tag{6.3}$$

4. The line 6: $B{=}A \otimes B$ performs computation shown in Equation 6.4 taking into account paths which start in set S_1 (possibly traversing many edges inside it) and then move to S_2 following single edge from S_1 to S_2. The result of this operation is presented in Equation

6.5. For instance, in updated block B we have distance between vertex 0 and 4 equal 3 what stems from including path $0-1-2-4$ (see Figure 6.1b). In this way, in set S_2 we obtain *bridge* vertices (marked with fatter border), for which the distance information includes also edges outside S_2. This allows for subsequent path length updates in the next steps.

$$(A \otimes B)_{i,j} = \min(A_{i,1} + B_{1,j}, A_{i,2} + B_{2,j}, \ldots, A_{i,n_1} + B_{n_1,j}), \tag{6.4}$$

$$A \otimes B = \begin{bmatrix} 3 & 2 & 1 & \infty \\ 2 & 1 & 2 & \infty \\ 1 & 2 & 1 & \infty \\ \infty & \infty & 1 & \infty \end{bmatrix}. \tag{6.5}$$

5. The computation made in line 7: $C{=}C \otimes A$ is presented in Equation 6.6. Here, we account for paths which start with single edge from S_2 to S_1 and then traverse sequence of edges inside S_1. In case of undirected graphs this update is symmetrical to one performed in line 6.
$$(C \otimes A)_{i,j} = \min(C_{i,1} + A_{1,j}, C_{i,2} + A_{2,j}, \ldots, C_{i,n_1} + A_{n_1,j}), \tag{6.6}$$

6. The operation $D{=}D \oplus C \otimes B$ (line 8) updates information about shortest paths between vertices in set S_2. Initially, the block D contains only S_2-internal edge weights. By performing $C \otimes B$ operation using previously processed blocks, we take into account paths which connect vertices from S_2 but contain only external edges joining *bridge* vertices. For instance, the path $4-2-6$ allows for setting distance between vertices 4 and 6 to 2 (see Figure 6.1c). In this step additional $\oplus$ operation is needed, as paths going outside S_2-induced edge set can be not the shortest ones.

7. The rest of R-Kleene algorithm steps can be explained in a similar way.

R-Kleene CUDA implementation

In this section we provide remarks on CUDA implementation of R-Kleene algorithm. Porting R-Kleene APSP to GPU was described in a recent work by Buluç [38]. Our contribution is the integration of CUDA program with *Graph Investigator* using JNI (*Java Native Interface*) and adjustments which improve its efficiency for undirected graphs and networks with bounded diameter. We use version 4.0 of CUDA library.

1. As the device code does not support recursion we use host recursion stack and invoke kernels implementing $\otimes$ and $\oplus$ operations.

2. The whole distance matrix is stored in the global device memory as a one-dimensional array in column-major order. In this scenario memory coalescing works well.

3. We employ two implementations of GPU MM. The standard MM kernel using thread blocks of size 16×16 and shared memory is called to process submatrices of sizes less than 256×256. For larger ones, we use optimized MM kernel by Volkov and Demmel [154], which takes advantage of smaller (16×4) thread blocks to maximize registers utilization.

4. By using shared memory in the parallel operation $\otimes$, the number of reads from global memory is minimized.

5. The single precision `float` 4B type is enough for computing shortest paths of weighted graphs, while for undirected graphs we use 2B `unsigned short`. This allows for better utilization of available GPU memory. The diameter of connected graph with n vertices is at most n, therefore `unsigned short` with maximal value 65635 is suitable for storing matrix with distance values bounded by 56281 (6GB memory). Nevertheless, using `short` type can reduce performance due to bank conflicts and `short` to `int` conversions. Thus, we use this type only for the graphs of size greater than 39796 (see Table 6.1).

6. For R-Kleene block sizes less than 64×64 sequential Floyd-Warshall algorithm is employed.

7. In case of undirected graphs the operation $A \otimes B$ is symmetrical to $C \otimes A$ and similarly $B \otimes D$ to $D \otimes C$. We use this relation to optimize program for this type of graphs. The MM kernel launches in lines 7 and 11 (see Listing 6) are skipped and the updates of submatrix C are performed inside kernels performing operations $A \otimes B$ and $B \otimes D$.

8. In order to increase bandwidth between host and device memory we use page-locked allocation on the host side.

9. The distance matrix updates presented in Equations 6.4 and 6.6 are performed in place, therefore the elements of the matrix can be modified by a thread from a different block during kernel execution. Nevertheless, this does not affect solution as the new value is always less or equal to the old one.

10. For well-connected graphs the branching in execution paths of tropical semiring multiplication kernels is lower (*min* function chooses one of two arguments much more frequently than the second one). Owing to lower branch divergence we obtain better *warp* efficiency for dense graphs.

Recursion allows for exploiting data locality, which is beneficial for stream processors. It brings cache efficiency improving computation performance.

6.3.2 Breadth-First Search

When, due to large graph size, a whole distance matrix cannot be stored in GPU memory, the GPU implementation of BFS algorithm [86,87] can be used to compute distances between given vertex and the rest of graph vertices. In this case the different representation of a graph in a form of packed edge list (one-dimensional array) is used [86]. Two kernels executed in a bulk synchronous parallel mode are needed to perform this task (see Listing 7). The synchronization of GPU threads (*computation phase*) between alternate kernel calls is obtained using CPU barrier (*communication phase*). The pseudo-code of the algorithm is presented in Listings 7, 8 and 9.

The vertices are visited in levels and boolean array $frontier$ shows all nodes processed at current level. At the beginning $frontier[i] == false$ except for one source vertex s. Kernel invocations stop if $frontier$ array contains only $false$ elements. Distance array *distance* stores lengths of shortest paths from the given vertex s to any other. The helper array $afrontier$ and additional kernel `FrontierUpdatekernel` are used to avoid read-write inconsistencies. The global array $frontier$ cannot be used in this step, because of possibility of read during update errors. The boolean matrix *visited* gives information whether a vertex was already visited.

As the number of threads executed in parallel depends on the number of vertices in currently visited level, the algorithm works more efficiently for dense graph.

Algorithm 7 `BFS`

```
1: while not terminate do
2:    terminate ← true;
3:    BFSkernel();
4:    synchronize();
5:    FrontierUpdatekernel();
6:    synchronize();
7: end while
```

Algorithm 8 `BFSkernel`

Input: $frontier$, $visited$ - n-element global boolean arrays
Input: s - vertex, initially $frontier[s] == true$
Output: $distance$ - n-element global integer array storing distances between s and the rest of vertices, initialized with zeros

```
1: tid ← getThreadId();
2: if frontier[tid] then
3:    frontier[tid] ← false;
4:    for all neighbor of tid do
5:       if not visited[neighbor] then
6:          distance[neighbor] ← distance[neighbor] + 1;
7:          afrontier[neighbor] ← true; {global helper frontier array}
8:       end if
9:    end for
10: end if
```

Algorithm 9 `FrontierUpdatekernel`

Input: $frontier$, $visited$, $afrontier$

```
1: tid ← getThreadId();
2: if afrontier[tid] then
3:    frontier[tid] ← true
4:    visited[tid] ← true
5:    afrontier[tid] ← false
6:    terminate ← false
7: end if
```

6.3.3 Linear algebra of graph matrices

A part of presented graph descriptors is defined with the use of linear algebra operations such as matrix-vector multiplication, matrix-matrix multiplication, *Laplace matrix* eigendecomposition. All of them can be computed using well-known CUDA libraries CUBLAS and CULA. Only available device memory limits the size of graph that can be analyzed this way.

6.4 Applications

In this section we show two examples of graph analysis with the use of GPU-enabled algorithms. First, we report GPU/CPU timings for tumor vascular network distance matrix computation. Next, we demonstrate comparison of several large networks obtained from synthetic models and *Stanford Large Network Dataset Collection* [11]. We employ B-matrices and several graph descriptors to show topological differences between those graphs. The descriptors presented in Section 6.1 were implemented using CUDA and integrated with *Graph Investigator* [58] via JNI. This allows for analyzing graphs of considerable size in an interactive way, which means that the computation of a descriptor for a single graph takes no longer than 3 seconds. In experiments we used two Nvidia GPUs: Tesla C2070 and GeForce GTX 260. Their technical brief is presented in Table 6.1. Tesla C2070 is a high end processing unit dedicated for general purpose computing (GPGPU), capable of delivering over 1Tflop single precision performance. In turn, GeForce GTX 260 is a powerful GPU which can be obtained with relatively low cost, and therefore it is found on many PCs. The size of available memory limits R-Kleene APSP algorithm to graphs with 39796 and 11484 vertices (Tesla C2070 and GeForce GTX 260 respectively, assuming `float` distance matrix). In case of BFS algorithm, which uses different graph representation in the form of packed adjacency list, we can analyze sparse graphs of size 10^7, however in this case the speedup is much smaller (GPU implementation 3-4 times faster).

Table 6.1: Specification of two GPUs used in experiments

	Tesla C2070	GeForce GTX 260
Number of cores	448	192
Memory	6GB	896MB
Memory bandwidth	144 GB/sec	111.9 GB/sec
Frequency of CUDA cores	1.15 GHz	1.24 GHz
SP Peak Performance	1.03 Tflops	715 Gflops
Architecture	Fermi	GTX 200
Compute capability	2.0	1.3
Max size of distance matrix[a]	39796×39796	11484×11484
Max size of distance matrix[b]	56281×56281	20060×20060

[a] `float` or `int`, 4B
[b] `short`, 2B

6.4.1 Tumor vascular networks

Understanding dynamics of tumor proliferation is vital for anticancer drug design and planning treatment [160, 161]. The process of tumor vascular network growth is governed by a set of pro- and anti-angiogenic factors. The tumor vasculature emerging from a complex interplay between spatial constraints, growth factor concentration fields and diffusion of inhibitors has a different structure than a normal arterio-venous blood vasculature (see also Section 3.4.1 and Section 5.3.2). It is heterogeneous, non-hierarchical and self-similar [162]. The non-trivial topology of such networks, possessing regions of high density, regions of low density and "holes" can be quantitatively captured using graph descriptors. For instance *efficiency* (A.1) allows for assessment of network ability to transport nutrients, oxygen and drug injections into tumor tissue.

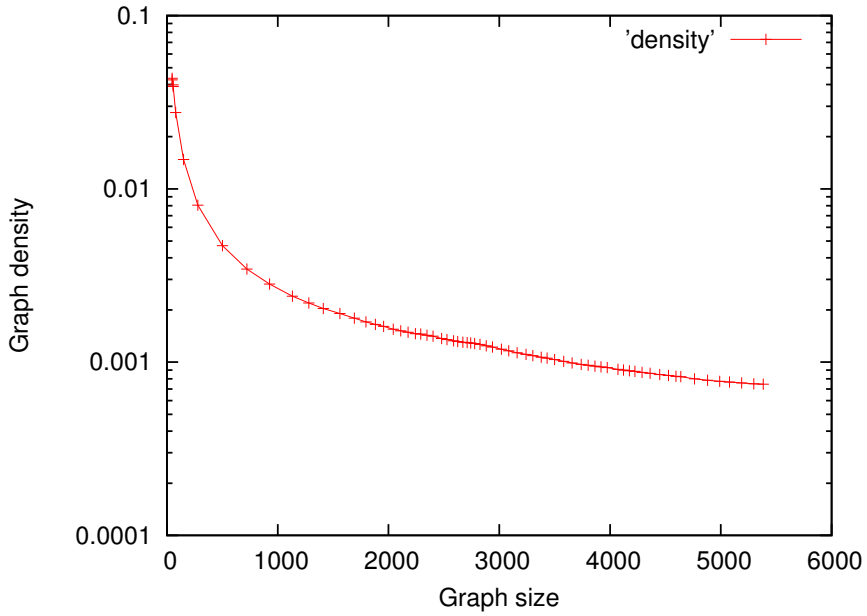


Figure 6.2: Densities of tumor vascular networks. The size of graphs grows with consecutive simulation steps.

The vast number of models of tumor-induced angiogenesis delivers output in a form of weighted or unweighted graphs. These graphs should be confronted with real vascular networks obtained from angiography or confocal microscopy. Thereby, we are able to validate model by comparing results of *in-silico* experiments with *in-vivo* ones. Evaluation of similarity between networks can be performed by packing various graph descriptors into feature vectors and embedding graphs into metric space [58]. Moreover, the descriptors can be also used as a "fingerprints" describing current state of tumor dynamics. The time of graph descriptors computation becomes critical as far as a number of combinations of model parameters that should be tested and the sizes of generated networks are concerned.

With a help of model described in [148], we generated 68 tumor vascular networks of size varying from 47 to 5400 . The sequence contains networks of different maturity and density (see Figure 6.2), captured at different simulation steps. As explained in Section 6.3.1, with CUDA implementation of R-Kleene algorithm, all-pair shortest-paths are computed faster for dense graphs than for sparse ones. We compared graph distance matrix computation times on Tesla C2070 GPU, GTX 260 and Intel i7-950 CPU. The CPU implementation was executed on single core, then the time of execution was divided by the number of available parallel threads (4×2 quad-core + hyper threading) in order to simulate perfect, parallel CPU implementation. The sample timings are depicted in Figure 6.3. The GPU-enabled analysis is about two orders of magnitude faster than CPU versions. This gives an opportunity to compare large graphs in a reasonable time. In the Figure 6.4 we show the effect of R-Kleene algorithm modification for undirected graphs. After restricting graph type and applying symmetrical distance matrix updates, we were able to accelerate GPU implementation on average 1.48 times for GeForce GTX 260 and on average 1.58 times for Tesla C2070. In Figure 6.5, we compare optimized GPU implementation executed on Tesla C2070 with perfectly-parallelized CPU implementation executed on Intel i7-950. The speedups for different graph sizes (averaged over 20 executions) are reported. The average speedup for vascular networks with more than 2000 vertices is 100.98. The more detailed discussion of the obtained results is presented in Section 6.5.

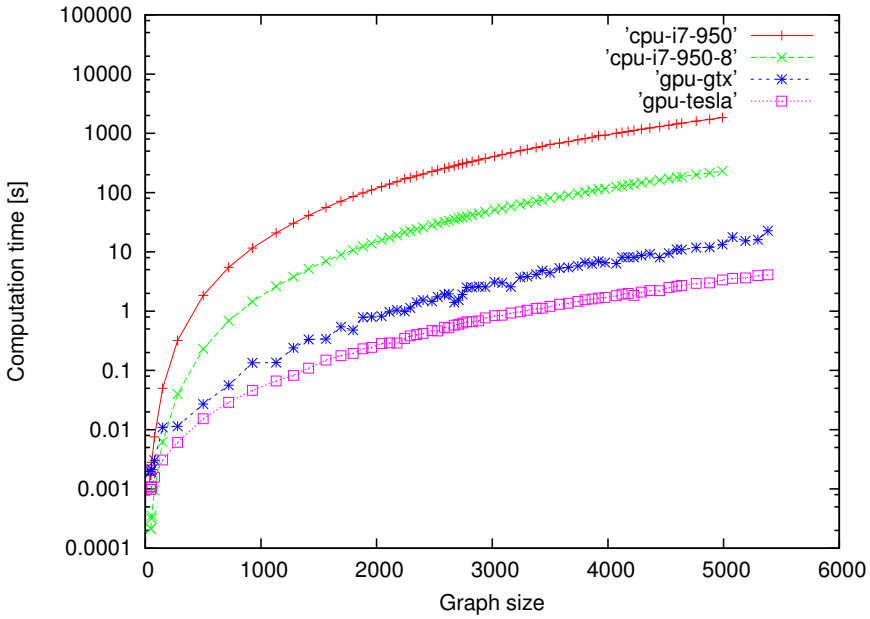


Figure 6.3: Comparison of graph distance matrix computation times for tumor vascular networks obtained from simulation [148]. Blue and violet lines correspond to CUDA implementation of R-Kleene APSP algorithm executed on GeForce GTX 260 and Tesla C2070 respectively. The red line - single core CPU execution of R-Kleene, *in-situ* implementation (i7-950). The green line shows CPU time divided by 8 to imitate parallel implementation exploiting perfectly 4 cores with hyper threading. Sequential version compiled using `gcc` 4.3.4 with `-O4` and `-ffast-math` optimization options.

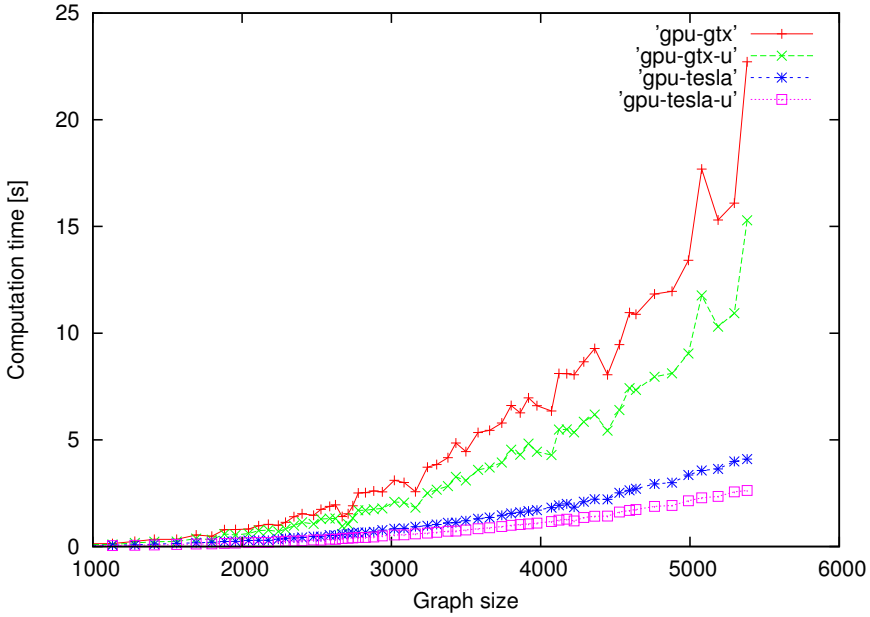


Figure 6.4: Comparison of two variants of R-Kleene algorithm executed on GPUs: Tesla C2070 and GeForce GTX 260. The timings labeled with -u suffix show the effect of symmetrical distance matrix updates. The average times of 20 executions are shown.

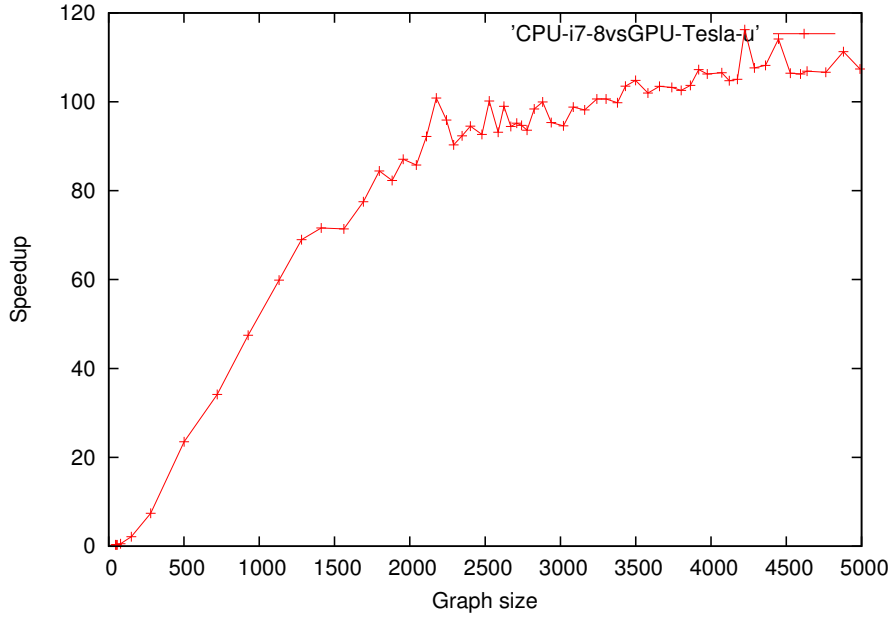


Figure 6.5: The speedup of GPU vs. CPU implementation of APSP for different graph sizes. The GPU version was implemented using symmetrical distance matrix updates. The CPU time divided by 8 as in previous experiments. The average times of 20 executions are presented. The average value of speedup for graph sizes above 2000 is 101.

6.4.2 GPU-enabled computation of graph invariants

This section reports experiment on networks of large sizes ranging from 21363 to 36863. First, we generated three artificial complex networks using Eppstein power-law model [61] (EP), Erdős-Rényi random graph model [62] (ER) and Kleinberg small-world model [101] (KL). Additionally, we analyzed three real-world graphs from *Stanford Large Network Dataset Collection* [11]. The nodes of Enron email network (EM) represent email addresses, while the edges model email exchanges between them. The edge exists if at least one email was sent from one address to another. We extracted GCC of this graph and used it in further analysis. The second real-world network (GN) represents Gnutella peer-to-peer file transfers. The vertices of this graph imitate hosts, while the edges stand for connections between them. We used Gnutella network shapshot from August 30 2002. The last graph sample is the GCC of Condense Matter Physics collaboration network constructed on the basis of arXiv e-prints (CM).

In the Table 6.2 we present several graph descriptors computed for the described set of large graphs with a help of CUDA implementation of R-Kleene APSP algorithm. Additionally, computation time of the distance matrix on Tesla C2070 GPU is reported (`float` type used). In Figure 6.6, we show vertex B-matrices of all six networks. As described in Chapter 4, B-matrices reflect structure of underlying graphs and can be used for their visual comparison. For instance, as illustrated by B-matrices in Figure 6.6 and confirmed by descriptor values, KL network differs from the others significantly. Although it has middle-range density, its efficiency is the lowest one. In addition, a small difference between diameter and radius indicates circle-like structure of this network. Relatively narrow histograms in B-matrix reflect generating mechanism of Keinberg model, which starts with 2D lattice and performs distance-dependent rewirings. High value of average path length, negatively correlated with efficiency, indicates inferior transport capabilities of KL network. This is because in the Kleinberg model

the probability of adding edge between two randomly selected vertices u and v is proportional to $d_G^{-2}(u,v)$, therefore long-range links forming shortcuts to different parts of a network appear rarely. In case of real-world networks, we observe similarity of B-matrices for two social networks EM and CM. Technological network GN appears to have different pattern of k-level degree frequencies, what reflects different generating mechanism of Gnutella connections. This conclusion is also supported by scalar descriptors from Table 6.2. The social networks have the highest values of density, efficiency and the lowest average shortest paths.

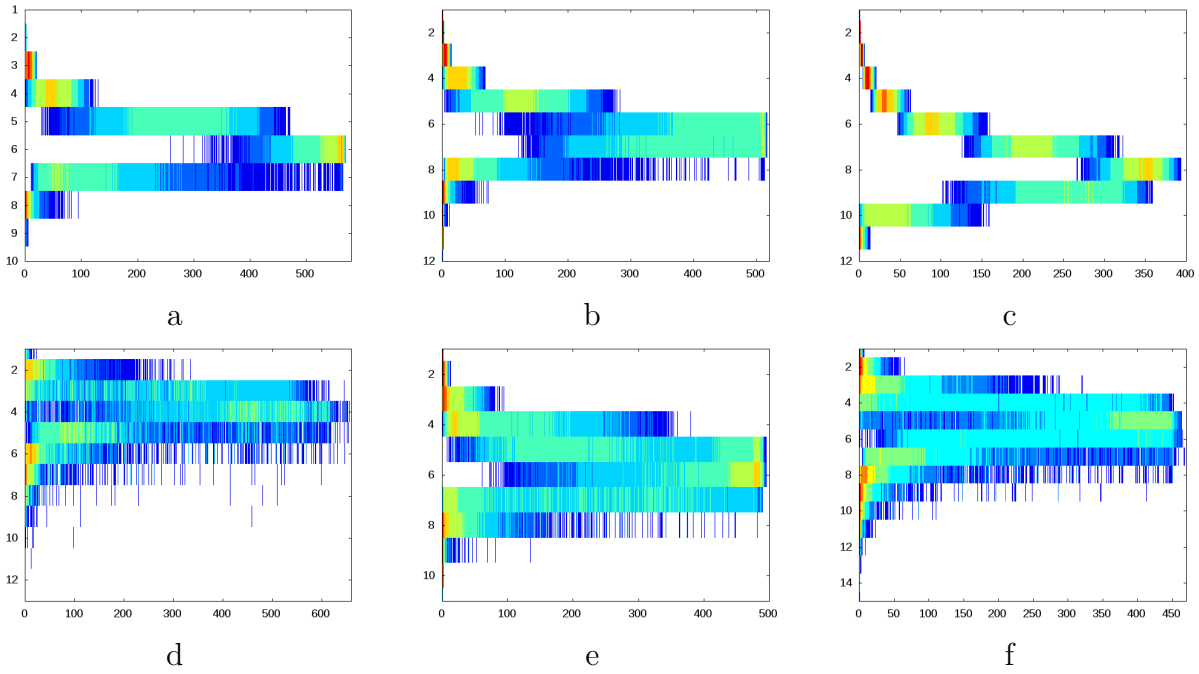


Figure 6.6: Vertex B-matrices of four large networks: a. Eppstein power-law network (EP), b. Erdős-Rényi network (ER), c. Kleinberg small-world (KL), d. email Enron network (EM), e. Gnutella p2p connections (GN) [135], f. condense matter physics collaboration network (CM) [105]. The networks d,e,f were obtained from *Stanford Large Network Dataset Collection* [11].

Table 6.2: Descriptors computed for sample large networks

	EP	ER	KL	EM	GN	CM
Vertices	35939	36863	28900	33696	36646	21363
Edges	120000	103287	86612	180811	88303	91286
Density	1.86e-4	1.52e-4	2.07e-4	3.19e-4	1.32e-4	4.00e-4
Diameter	10	12	12	13	11	15
Radius	7	8	10	7	7	8
Efficiency	0.0892	0.0813	0.0665	0.1313	0.0899	0.0990
Average path length	5.74	6.31	7.78	4.03	5.74	5.35
Wiener index	3708445310	4287364508	3248051522	2285058869	3860710021	1221244749
Norm Wiener index	4.79e-4	5.14e-4	8.07e-4	3.58e-4	4.71e-4	7.52e-4
GPU Time[a][s]	747.66	797.48	386.13	575.16	796.91	157.43

[a] Distance matrix computation including device $\leftrightarrow$ host memory transfers, Tesla C2070, timing using CUDA event API

6.5 Discussion

We have presented how to accelerate graph comparison and visualization using GPU implementations of all-pair shortest-paths and breadth-first-search algorithms. It was shown that highly-optimized variants of matrix multiplication allow for increasing a size of graphs analyzed interactively by two orders of magnitude. The contribution of presented research is enhanced *Graph Investigator* application capable of performing descriptor calculations for large graphs. Only available GPU memory limits the size of networks that can be analyzed on the basis of shortest-paths distance matrix.

In the experimental section we reported significant, two orders of magnitude speedup (Tesla C2070, GTX 260) over sequential implementation of R-Kleene executed on a single-core of i7-950 processor. Here, we would like to give several remarks on this result. As described in the work of Lee et al. [104], the fair comparison of GPU and CPU performance requires applying appropriate platform-specific optimizations, especially on the CPU side. The authors argue, that rigorous performance tuning can narrow the gap between GPU and CPU to less than 10 times, depending on a kernel type and a number of cores ratio. They compare efficiency of high-throughput kernels executed on i7-960 3.2GHz quad-core processor and GeForce GTX 280 with 240 cores using CUDA 2.3. In our study we used Tesla C2070 with 448 computing cores, i7-950 quad-core 3GHz processor, CUDA 4.0 and sequential code compiled with level 4 optimization. Here, the difference of computing power is more noticeable, what together with the influence of the newest Fermi GPU architecture and performance enhancements in CUDA 4.0 allows for obtaining significant speedup. More careful optimization of the CPU code could decrease difference in APSP computation time. Nevertheless, from the perspective of *Graph Investigator* application usability, taking advantage of GPU-enabled algorithms improves framework interactivity, no mater whether the speedup is 100× or 80×.

The main drawback of presented approach is the limitation for the size of analyzed graph implied by the need for storing whole distance matrix in GPU memory. This can be partially overcome by using `short` type or multi-device programming and unified virtual address space [13]. Still, the GPU memory and host RAM are limited resources and finally we will get to the point where distance matrix should be divided into more than 4 blocks and disk↔RAM↔GPU data transfers together with distance-submatrices processing will have to be performed to obtain all-pair shortest-paths for a large graph. This is the direction for the future work.

Chapter 7

Dissertation summary

In this work we addressed the problem of graph comparison and its applications in quantitative analysis of structured data derived from complex networks and pattern recognition areas. The author's contribution to the field is described in the following list.

1. Novel graph embedding method based on distance k-graphs invariants

   Graph pattern vectors should be invariant under graph isomorphism. In the existing graph embedding algorithms, this prerequisite is enforced with the use of permutation invariant functions or transformations operating on a sequence of elementary features whose order depends on initial vertices ordering implied by the graph representation used. For instance, the spectrum of Laplace matrix, being invariant under similarity transformation, is frequently employed as an informative set of graph characteristics. In this work we proposed a different approach, which takes advantage of an ordered set of graphs derived from a source graph on the basis of vertex-vertex dissimilarities. We obtained decomposition of an original graph into sequence of distance k-graphs and used invariants of those graphs to form feature vectors reflecting the structure of the source graph. This approach is general, in that we can use different vertex-vertex metrics or apply it for weighted graphs as well. In particular, we studied aggregated degree histograms of distance k-graphs derived from the shortest path metric, which form so-called B-matrices of a graph. This type of graph invariants can be computed efficiently using BFS or APSP algorithms constituting information-rich basis for explicit graph embedding. We have shown, how the structure of vertex and edge B-matrices is related to the structure of underlying graph and how this correspondence can be utilized in the generation of a lower-dimensional feature vectors. Experiments on clusterization and classification with synthetic and real-world structured data revealed that the new descriptors allow for distinguishing graphs with non trivial structural differences. Moreover, they appear to outperform descriptors based on the Laplace matrix decomposition, being at the same time more effective computationally.

2. *Graph Investigator* application being a self-contained software tool for analysis of clusters of graphs

   We created a publicly available Java program which allows for quantitative analysis and comparison of grouped and non-grouped network datasets. The application is capable of computing more than 100 parametrized graph descriptors and performing cluster analysis. The B-matrix-based visualization and matching are also available. We have demonstrated how to use *Graph Investigator* for topological analysis of vascular networks and validation of tumor-induced angiogenesis models. The presented pure-structural, non-spatially

constrained approach allows for analysis of tumor vasculature in case-invariant, universal manner.

3. Improving interactivity of *Graph Investigator* program with the use of GPU-enabled graph algorithms, which accelerate computation of distance-based graph descriptors

   In order to enhance the usability of *Graph Investigator* application for large and medium-size graphs, we implemented APSP algorithm in CUDA and harvested computational power of GPU. Primarily, we used recently described R-Kleene algorithm for APSP problem and performed optimizations for undirected graphs. This step allowed us to obtain 100× speedup over single-core CPU implementation. As many graph descriptors take advantage of the information about shortest paths, using GPU-implementation of APSP and JNI we could significantly improve response times of *Graph Investigator* in graph comparison tasks.

7.1 Conclusions

In this dissertation we stated two major theses (Section 1.2), which were critically assessed in the experimental sections of the *Contribution* part.

As shown in Chapter 4, vertex-vertex dissimilarity measures, such as shortest-paths or commute-times, can be used to generate ordered set of distance k-graphs. We proposed to employ several elementary invariants of those graphs for extraction of expressive graph features. Two types of distance k-graphs were studied: vertex distance k-graph and edge distance k-graph. In particular, we investigated degree distributions of distance k-graphs forming B-matrix representations invariant under graph isomorphism. We introduced six graph descriptors based on B-matrix that is $D^{\star}_{long}$ (row-aggregated B-submatrix), $D^{\star}_{rstd}$ (relative standard deviations of B-matrix rows), $D^{\star}_{ent}$ (Shannon entropies of B-matrix rows), $D^{\star}_{avgd}$ (differences between row-averages of B-matrix), $\mu^{\star}$ (row average values) and $\sigma^{\star}$ (row standard deviations). The experiment testing the relation between graph edit distance and Euclidean distance derived from those embeddings showed that our pattern vectors follow graph edit distance nearly linearily. This confirms stability of the new embedding method. In the tests on unsupervised learning using sythetic data (Section 4.3.2) and metabolic networks (Section 3.3.1) we demonstrated that our descriptors outperform reference, state-of-art explicit graph embedding methods. This is also true for not-parameterized PCA dimensionality reduction, what makes our approach practical in many applications as there is no need to estimate values of any parameters. It was shown that long pattern vector $D^{\star}_{long}$ is particularly useful in graph comparison, yielding satisfactory results in all experiments. We futher demonstrated the relevance of B-matrix-based features in classification of satellite photos (Section 4.3.4). In this experiment we received on average 10% better classification accuracy than for reference descriptors. The mutagenicity dataset from benchmark IAM database was utilized for testing purely-structural supervised learning with $D^{\mathcal{E}}_{long}$ pattern vector. After performing feature selection on $D^{\mathcal{E}}_{long}$ descriptor, we obtained good classification rate with relatively low computational cost. These results are only slightly worse than the ones got for computationally expensive method based on graph edit distance, which takes into account both structure and attributes. The experiments mentioned above support thesis stated in Section 1.2, point 1.

In Chapter 5 we described *Graph Investigator* application, developed as a software contribution of this work. The program is a self-contained framework for analysis of graphs using

methods of statistical pattern recognition. *Graph Investigator* allows for computation of over 100 graph descriptors, including novel invariants described in Chapter 4. We demonstrated how our application can be utilized in analysis of vascular networks. The sample use cases include investigation of brain vasculature (Section 5.3.1) and tracking evolution of tumor vasculature modeled using cellular automata paradigm (Section 5.3.2). In Chapter 6 we showed how to improve interactivity of *Graph Investigator* application by implementing APSP and BFS graph algorithms in CUDA and utilizing them inside Java program via JNI. In particular, the enhanced, GPU-enabled version of APSP R-Kleene algorithm, executed on Nvidia Tesla C2070 allowed us to obtain 100× speedup in the computation of graph distance matrix. This is benefitial as many descriptors which take advantage of distance information (see Section 6.1) can be computed more efficiently, for instance in time less than 3 seconds for graphs of size 10^4. The research described in this part of the dissertation supports thesis stated in Section 1.2, point 2.

7.2 Relevance of results

The new method for graph feature vector generation presented in our work was compared with a set of descriptors from theory of complex networks and the state-of-art spectral method based on heat kernel matrix. It was shown in the work by Xiao and Hancock [166] (2009) that those descriptors outperform previously known spectral embeddings [107, 164], being more expressive and allowing for navigation between local and global features using time parameter. By selecting most robust heat kernel descriptors D_{hkc} and D_{hkcc} (see A.15) we aimed to perform fair comparison between our method and spectral embeddings. Focusing on descriptors derived from B-matrices, we proposed a method, which is less expensive computationally than eigendecomposition, particularly for sparse graphs being more frequent in real-world applications. Moreover, graph distance computation can be performed with a single precision or even using integer types. Also numerical stability is not an issue in APSP algorithms. The graph invariants based on distance k-graphs allow for generation of high-dimesional feature vectors forming an interesting alternative for other graph embedding methods.

Although there exists a number of software graph analysis tools (see Section 5.1), we decided to develop *Graph Investigator* application to have at hand the program which would allow for a comparison of groups of graphs and flexible addition of new graph descriptors. The group-based comparison is a feature not present in related applications. In order to get a self-contained framework, we also added functionalities like visualization, import/export, conversion, component extraction, etc. Nevertheless graph embedding and analysis using methods of statistical pattern recognition was our main goal. The application was particularly useful in the study of angiogenesis models built by Topa [148] and Wcislo [161] but it can also support the analysis of different structural datasets (see Section 4.3).

After integrating *Graph Investigator* with CUDA-enabled APSP R-Kleene algorithm (described by Buluç in 2010, [38]) via JNI we got a more robust tool. With this improvement our application is capable of computing distance-based graph descriptors for large graphs in an interactive way.

7.3 Discussion

The comparison of structured objects using explicit graph embedding is a convenient methodology for building a bridge between structural and statistical pattern recognition, nevertheless it also brings certain pitfalls. A part of them were addressed in Section 2.2. Here, we would like to discuss the weaknesses specific to matching based on distance k-graphs and limitations of implemented software.

As the number of non-zero rows in B-matrix depends on a graph diameter, the presented approach for graph comparison is diameter-centered. The B-matrices of graphs with distant diameters differ significantly thus dissimilarity ranks computed on the basis of our descriptors will be high. In a general case it seems reasonable and more appropriate than building pattern vectors of dimensionality dependent on a number of vertices. For a significant part of graphs the diameter grows slowly with the number of vertices (e.g. for *small-world* networks or Erdős-Rényi graphs $diam(G) \sim \log(n)$), therefore our method is less vulnerable to different-size effects. Nevertheless, to extract topological features common for topologically similar graphs with much different diameters, e.g., small 2D mesh and large 2D mesh, we need to adjust bin sizes and perform normalization (see Section 4.2), obtaining coarse-grained B-matrices. Selection of a relevant subset of B-matrix features is also an important issue (especially for high-dimensional $D^{\star}_{long}$ descriptor). In case of supervised learning this can be achieved using validation set (Section 4.3.5) but for unsupervised learning one needs to use dimensionality reduction techniques or apply additional knowledge such as selecting lower-rows of B-matrix for more local features.

The greatest limitation of *Graph Investigator* application is time-expensive computation of more elaborate graph descriptors for large or even medium-size graphs. We tackled this problem by employing CUDA-enabled APSP algorithms, what brought significant performance boost to our application. Nevertheless, the improvement works only for graph descriptors based on distance information. Due to limited GPU memory we are able to use this algorithm only for graphs with number of vertices less than 56281. The larger datasets shall be analyzed in a different way and this problem will be addressed in future works.

7.4 Future work

We plan to perform futher study of distance k-graphs properties including application of other vertex-vertex dissimilarity measures (not necessarily metrics) such as commute-time, f-communicability or longest-paths. This, together with the use of different distance k-graph invariants like *clustering coefficient*, can bring new robust graph descriptors. Also generation of vertex invariants and edge invariants on the basis of ordered set of k-shells is worth future investigation. As B-matrices can be used for visual assessment of graph structural characteristics, their application for visual evaluation of clustering based on kNN graphs seems promising.

The *Graph Investigator* application will be further developed, hopefully with a contribution and feedback from many users. There are also plans to integrate *Gin* program with an interactive tool for planning cancer treatment as a part responsible for simulation results validation and quantitative analysis of vasculature [160].

Appendices

Appendix A

Graph descriptors

A.1 Efficiency

Efficiency [30] is the harmonic mean of geodesic lengths over all couples of nodes. The normalization factor $n(n-1)$, proportional to number of node couples, ensures that it lies within range $[0, 1]$.

$$Ef(G) = \frac{1}{n(n-1)} \sum_{u,v \in V(G), u \neq v} \frac{1}{d_G(u,v)} \tag{A.1}$$

Efficiency measures the traffic capacity of a network and reflects its parallel-type transfer ability.

A.2 Wiener index

Wiener index, proposed in [163], is a sum of lengths of the shortest paths between every pair of vertices in a given graph G.

$$W(G) = \frac{1}{2} \sum_{u \in V(G)} \sum_{v \in V(G)} d_G(u,v) \tag{A.2}$$

It is used in chemistry as a topological index of molecule, reflecting its branching and correlated with its van der Waals surface. Wiener index reaches minimal value $\frac{1}{2}n(n-1)$ for a complete graph and maximal value $\frac{1}{6}(n^3-n)$ for a path graph, where n is the number of vertices. Using the latter formula we can normalize Wiener index as follows

$$W_n(G) = \frac{6W(G)}{n^3-n}, \tag{A.3}$$

obtaining $W_n(G) \in [0, 1]$.

A.3 Clustering Coefficient

This graph descriptor measures neighborhood connectivity used to indicate small-worldliness of a network [159]. The clustering coefficient of vertex v defined as follows:

$$C(v) = \frac{2|\{e_{ij}\}|}{k_v(k_v-1)}, \tag{A.4}$$

where $i, j \in N_v$ and $e_{ij} = \{i, j\} \in E(G)$, is a ratio of number of connections between neighbors of vertex v (denoted by $|\{e_{ij}\}|$) to number of links that could possibly exist between them, i.e., $k_v(k_v - 1)/2$. Clustering coefficient quantifies local topology of a graph, reflecting how close the neighborhood of a given vertex is to form a complete graph. The clustering coefficient for a graph G with n vertices is an average of clustering coefficients for each vertex.

$$C(G) = \frac{1}{n} \sum_{v \in V(G)} C(v) \tag{A.5}$$

The clustering coefficient $C(G)$ is related to the number of triangles (closed triplets) in a graph. The above formula is undefined for vertices of degree 1 or 0. In these cases $C(v)$ is usually set to 0, however as pointed out in [94], such procedure may lead to biased assessments of neighborhood clustering when undefined values dominate average. Therefore, additionally we provide different graph clustering coefficient computed using vertices with more than one neighbor [94]:

$$C'(G) = \frac{1}{n'} \sum_{v \in V(G): k_v > 1} C(v), \tag{A.6}$$

where n' is the number of vertices of valence greater than 1.

A.4 Weighted clustering coefficient

Generalized clustering coefficient introduced in [95] quantifies local neighborhood connectivity and is used for assessment of *small-worldliness* of a weighted network.

$$WCC(G) = \frac{1}{n} \sum_{k=1}^{n} \frac{(R^3)_{k,k}}{(e^T r_k)^2 - \|r_k\|_2^2}, \tag{A.7}$$

where R is matrix of non-negative edge weights, r_k denotes k-th row of matrix R and e is an all-ones vector.

A.5 Average path length

$$Apl(G) = \frac{2}{n(n-1)} \sum_{u,v \in V(G)} d_G(u, v), \tag{A.8}$$

A.6 Graph diameter

The diameter of graph G, $diam(G)$, is the maximum length of distance between any two vertices in the graph.

$$diam(G) = \max_{u,v \in V(G)} d_G(u, v) \tag{A.9}$$

This measure reflects the density of graph connections, achieving its maximal value for paths, and minimal for cliques.

A.7 Subgraph Count

The l^{th}-order Subgraph Count for a graph G, ${}^{l}SC(G)$, is a number of connected subgraphs with k edges.

$$ {}^{l}SC(G) = |\{H = (V(H), E(H)) \subset G \colon |E(H)| = l \wedge diam(H) < +\infty\}| \tag{A.10} $$

This descriptor shows how many specified subgraphs with k edges occur in the graph. It can be normalized as follows

$$ {}^{k}SC_n(G) = \frac{{}^{k}SC(G)}{{}^{k}SC(K)}, \tag{A.11} $$

where K is a complete graph (clique) with n vertices. The common descriptors, derived from generic SC descriptors are Platt Index [121] ($l = 2$) and Gordon-Scantleburry Index [79] ($l = 3$). They reach their maximal values for cliques.

A.8 Betweenness Centrality

Centrality is a measure of relative importance of graph vertex according to given criteria. Betweenness measures are a kind of centrality measures used often in the analysis of social networks or citation networks. They tend to evaluate the influence of each vertex on spreading information over the graph. Shortest-path betweenness is a widely used centrality measure defined as a fraction of the shortest paths between pairs of vertices in a graph that pass through given vertex [71], [35], i.e.,

$$ BC(v) = \sum_{\substack{s \neq v \neq t \in V(G) \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{A.12} $$

where $\sigma_{st}(v)$ is the number of the shortest paths from s to t which pass through v and σ_{st} denotes the total number of the shortest paths from s to t. It measures to what extent a given vertex is needed by other vertices to transfer information through the shortest paths. The betweenness centrality for a graph G with n vertices is an average of betweenness centrality for each vertex.

$$ BC(G) = \frac{1}{n} \sum_{v \in V(G)} BC(v) \tag{A.13} $$

A.9 Random Walks Betweenness Centrality

The shortest path betweenness described above aims at evaluating the influence of each vertex on spreading information over network through geodesic routes. This model assumes that information does not move through other non-optimum paths, what in many cases is not true. In [117], Newman proposed a general betweenness measure, which takes into account all paths between any two nodes yet giving greater weight to shorter paths. This betweenness can be defined either by an analogy to current flow in an electrical network or by random walks. These two definitions are equivalent. In the first one, we treat a graph as an electrical circuit with unit resistances on edges and average the current flowing through given vertex out, involving all pairs of current-in and current-out. In the latter definition, one calculates the mean number of passages the random walker travelling between any pair of nodes makes through a given vertex. The random walks betweenness centrality $RWB(G)$ is an average of random walks betweenness centrality for each vertex.

A.10 Information of vertex degrees

$$I_{vd}(G) = \sum_{v \in V(G)} (k_v \cdot \log_2 k_v) \tag{A.14}$$

Reflects connectivity and topological complexity in terms of number of branches, cycles, cliques etc.

A.11 Estrada Index

The Estrada Index is a spectral descriptor, defined as a trace of the exponential adjacency matrix. It quantifies the content of subgraphs in the graph [63] taking into account closed walks of lengths 2 to ∞, encoded on the diagonal of powered adjacency matrix.

$$EE(G) = \sum_{i=1}^{n} \exp(\lambda_i) = tr(e^{A_G}), \tag{A.15}$$

where λ_i is i-th eigenvalue of adjacency matrix A_G and e^{A_G} is defined as follows

$$e^{A_G} = I + A_G + \frac{A_G^2}{2!} + \frac{A_G^3}{3!} + \dots \tag{A.16}$$

A.12 Density

The density descriptor is a ratio of number of exiting edges to the number of all possible edges in a given graph G.

$$density(G) = \frac{2m}{n(n-1)} \tag{A.17}$$

A.13 Volume

The volume of a graph is a sum of degrees for all vertices.

$$vol(G) = \sum_{v \in V(G)} k_v \tag{A.18}$$

A.14 Cheeger constant

For a graph $G = (V(G), E(G))$ and subset of vertices $U \subset V(G)$, the *normalized cut* is defined as follows

$$\phi(G, U) = \frac{e(U, U^C)}{\min(vol(U), vol(U^C))}, \tag{A.19}$$

where U^C denotes the complement of U in graph G, $e(U, U^C)$ is the number of edges with one vertex in set U and second vertex in set U^C, while $vol(\dots)$ denotes volume of induced subgraph. The Cheeger constant or isoperimetric number of the graph is defined as minimum achievable ratio $\phi(G, U)$, that is

$$\phi_G = \min_{U \subset V(G)} \frac{e(U, U^C)}{\min(vol(U), vol(U^C))}. \tag{A.20}$$

A.15 Heat kernel invariants

Heat kernel - the fundamental solution of heat equation associated with the Laplacian (see Equation A.21 and Equation A.22, ϕ_i denotes eigenvector associated with an eigenvalue λ_i) allows for construction of valuable graph descriptors that additionally can be scaled by the time parameter [166]. This enables to navigate between local properties (low values of t) and global properties (high values of t) of a graph.

$$\frac{\partial h_t}{\partial t} = -\mathcal{L}_G h_t \tag{A.21}$$

$$h_t = \exp(-\mathcal{L}_G t) = \sum_{i=1}^{n} \exp(-\lambda_i t)\phi_i \phi_i^T, \tag{A.22}$$

$$h_t(u, v) = \sum_{i=1}^{n} \exp(-\lambda_i t)\phi_i(u)\phi_i(v). \tag{A.23}$$

Graph characteristics generated on the basis of heat kernel [166] include heat kernel content invariant:

$$D_{hkc}(t) = \sum_{u \in V} \sum_{v \in V} \sum_{k=1}^{n} \exp(-\lambda_k t)\phi_k(u)\phi_k(v), \tag{A.24}$$

and heat kernel content coefficients invariant:

$$D_{hkcc}(m) = \sum_{k=1}^{n} \left\{ \left(\sum_{u \in V} \phi_k(u) \right)^2 \right\} \frac{(-\lambda_k)^2}{m!}, \tag{A.25}$$

where (λ_k, ϕ_k) is k-th eigenpair of normalized Laplace matrix of graph G.

Appendix B

Graph Investigator

Here we present the rest of network descriptors available in *Graph Investigator*, not included in Appendix A. First table contains scalar descriptors assigned to graph. Next, we list vertex and edge descriptors intended for local topology analysis. Statistical moments of these metrics can be used to generate general graph descriptors. *Graph Investigator* enables to compute mean, standard deviation, skewness and kurtosis of vertex or edge descriptors.

B.1 General graph descriptors

Descriptor	Remarks
Randić Connectivity Index [128]	$\chi(G) = \sum_{(v,w)\in E(G)} (k_v k_w)^{-1/2}$, $\|E(G)\|\chi(G) \leqslant \frac{\|E(G)\|}{\|V(G)\|-1}$ Connectivity measure derived from chemical graph theory
General Connectivity Index [98]	${}^{l}\chi(G) = \sum_{P_l \subset G} \left(\prod_{w\in V_P} k_w\right)^{-1/2}$, where P_l denotes path of length l, and V_P all vertices that belong to this path
Zagreb Index M_1 [84]	$M_1(G) = \sum_{v\in V(G)} (k_v)^2$
Zagreb Index M_2 [84]	$M_2(G) = \sum_{e\in E(G)} w_e$, where w_e is the weight of edge e. For unweighted graphs $\forall_{e\in E(G)}\ w_e = 1$
Modified Zagreb Index ${}^{m}M_1$ [119]	${}^{m}M_1(G) = \sum_{v\in V(G)} (k_v)^{-2}$
Modified Zagreb Index ${}^{m}M_2$ [119]	${}^{m}M_2(G) = \sum_{e\in E(G)} (w_e)^{-1}$
Total Adjacency Index	$A(G) = \sum_{v\in V(G)} k_v = 2\|E(G)\|$
Modified Total Adjacency Index	${}^{m}A(G) = \sum_{v\in V(G)} (k_v)^{-1}$
B Index [31]	$B(G) = \sum_{v\in V(G)} \frac{k_v}{d_v}$, where d_v is the vertex distance (see B.2) for vertex v

Density of edges	$Den(G) = \frac{2m}{n(n-1)}$, where $m = \lvert E(G)\rvert$
Radius of graph	$r(G) = \min_{v \in V(G)} e_v$, where e_v is eccentricity (see B.2) of vertex v
Total Walk Count [137]	Counts all paths of all lengths in the graph and depends on the size, cyclicity and branching of the graph, quantifying property called *labyrinthicity*.

B.2 Vertex descriptors

Descriptor	Remarks
Vertex distance	Sum of distances between v and all other vertices from graph G, $d_v = \sum_{w \in V(G)} d_G(v, w)$
Eccentricity	Maximum distance between vertex v and any of the remaining graph vertices, $e_v = \max_{w \in V(G)} d_G(v, w)$
Vertex B Index	$b_v = \frac{k_v}{d_v}$, where d_v is vertex distance for vertex v
Randić Shortest Path Index	${}^m\chi(G) = \sum_{P \in \mathcal{P}_m} \left(\prod_{u \in V(P)} k_u \right)^{-1/2}$, where $\mathcal{P}_m = \{P(v, w) \subset G \colon d(P(v, w)) = m\}$
Page Rank [80]	Vertex importance measure based on graph random walks model. Determines probability of turning up in vertex v after long-time random walk
Hubs and Authorities measure [100]	Evaluates authority of a vertex on the basis of link structure
Local efficiency [30]	$E_{loc}(v) = E(G_v)$, where G_v is a subgraph of neighbors of v and $E(\ldots)$ is graph efficiency.
Closeness	Mean distance to any other vertex, $c_v = \frac{1}{n-1} \sum_{u \in V(G), u \neq v} d_G(u, v)$

B.3 Edge descriptors

Descriptor	Remarks
Edge connectivity	$EConn(e) = k_v \cdot k_w$
Range of edge [158]	$g(e) = d_{G'}(v, w)$, where $G' = (V(G), E(G) \backslash \{e\})$
Edge frequency [77]	The edge frequency for edge e, is a number of shortest paths, which contain edge e
Edge betweenness	Measures relative importance of an edge in shortest-path transfer through graph edges.

Appendix C

Clustering validation indices

C.1 Davies-Bouldin Index

Davies-Bouldin Index [83] is defined as follows

$$DB = \frac{1}{m}\sum_{i=1}^{m} \max_{j=1,\ldots,m;j\neq i} d_{ij}, \tag{C.1}$$

$$d_{ij} = \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}, \tag{C.2}$$

where m is a number of clusters, σ_i is an average distance of all points in cluster i to their cluster center c_i and $d(c_i, c_j)$ is a distance between cluster centers c_i and c_j. Davies-Bouldin index lies within range $[0, \infty)$ with small values indicating good clustering.

C.2 C Index

The C Index [83] is defined as a quotient

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}, \tag{C.3}$$

where S is the sum of distances over all k pairs of points from the same cluster, S_{min} is the sum of k smallest distances for all pairs of points, whereas S_{max} is the sum of k greatest distances for all pairs of points; $C \in [0, 1]$ with small values indicating good clustering.

C.3 Rand Index

The Rand Index [127] is used to evaluate similarity between two clusterings. Particularly, in this work we employ it for validation of unsupervised learning results. To this end we compare real cluster memberships with a results of k-NN algorithm by measuring fraction of agreements to disagreements in cluster assignments. If over a half of neighborhood of given object i belongs to the same class as i, we count it as agreement.

$$R = \frac{L}{L + P}, \tag{C.4}$$

where L is the number of agreements and P is number of disagreements in the cluster assignments; $R \in [0, 1]$ with values close to 1 indicating good clustering.

Appendix D

Commute Time

The *commute time* is a vertex-vertex metric related to random walks on a graph. The notion is also linked to a diffusion process on a graph and heat kernel matrix (see A.15). Let us first recall definition of *hitting time* from a work by Qiu and Hancock [123].

Definition D.0.1 *The hitting time $HT(u, v)$ of a random walk on a graph is the expected number of steps before node v is visited, starting from node u.*

Definition D.0.2 *The commute time $CT(u, v)$ is the expected number of steps for a random walker to travel starting from node u, reaching node v and coming back to u.*

$$CT(u, v) = HT(u, v) + HT(v, u) \tag{D.1}$$

Let (λ_k, ϕ_k) be the k-th eigenpair of normalized Laplace matrix of graph G ($\mathcal{L}_G$). The normalized Green's function, being the pseudoinverse of the normalized Laplacian $\mathcal{L}_G$, can be expressed as follows

$$\mathcal{G}(u, v) = \sum_{i=2}^{n} \frac{1}{\lambda_i} \phi_i(u)\phi_i(v). \tag{D.2}$$

As presented in [43], the *hitting time* can be computed using following expression

$$HT(u, v) = \frac{vol(G)}{k_v} \mathcal{G}(v, v) - \frac{vol(G)}{\sqrt{k_u k_v}} \mathcal{G}(u, v), \tag{D.3}$$

after taking the above equations, we can obtain formula for commute time

$$CT(u, v) = vol(G) \sum_{i=2}^{n} \frac{1}{\lambda_i} \left(\frac{\phi_i(u)}{\sqrt{k_u}} - \frac{\phi_i(v)}{\sqrt{k_v}} \right)^2. \tag{D.4}$$

The commute time metric is used in dimensionality reduction algorithms and clustering graphs [123].

Notation

$G = (V(G), E(G))$ – graph with vertices set $V(G)$ and edges set $E(G)$

$\hat{V}(G)$ – subset of $V(G)$ containing vertices with non-zero degree

$G_{n,m}$ – graph with n vertices and m edges

A_G – adjacency matrix of graph G

M_G – incidence matrix of graph G

L_G – combinatorial Laplace matrix of graph G

$\mathcal{L}_G$ – normalized Laplace matrix of graph G

n – order of graph (number of vertices)

m – size of graph (number of edges)

$e_{uv} = \{u, v\} \in E(G)$ – undirected edge (unordered pair of vertices)

$u \sim v$ – adjacent vertices u and v

$d_G(u, v)$ – length of the shortest path between u and v (distance between vertex u and v)

$d_G^{\mathcal{E}}(w, e_{uv})$ – the distance from a vertex v to an edge e_{uv}

N_v – neighborhood of vertex v (the set of vertices adjacent to vertex v)

k_v **or** $deg(v)$ – degree (valence) of vertex v (a number of edges which join v with its neighbors)

$\overline{k}(G)$ – mean vertex degree of graph G

$G \simeq H$ – graph G is isomorphic to graph H

$G \overset{\alpha}{\simeq} H$ – graph G is isomorphic to graph H wiht α denoting isomorphism

$G \not\simeq H$ – graph G is not isomorphic to graph H

$\mathbb{D}_G$ – the set of all descriptors of graph G

$\mathbb{G}_{n,m}$ – the set of all graphs with n vertices and m edges

$\mathbb{G}$ – the set of all graphs

$Iso(G)$ – the set of all graphs isomorphic to G

$\Delta_A(G)$ – spectrum (set of eigenvalues) of graph G adjacency matrix A

$\Delta_L(G)$ – spectrum (set of eigenvalues) of graph G Laplace matrix L

$X(k,:)$ – k-th row of matrix X

$X(:,k)$ – k-th column of matrix X

$(\mathbb{S}, \oplus, \otimes, \mathsf{0}, \mathsf{1})$ – semiring, where $\oplus$ and $\otimes$ are binary operations on set $\mathbb{S}$,

$\mathsf{0}$ is neutral element of $\oplus$ operation and $\mathsf{1}$ is neutral element of $\otimes$ operation

$G_k^{\mathcal{V}}$ – distance k-graph derived from graph G

$G_k^{\mathcal{E}}$ – edge distance k-graph derived from graph G

G^k – k-th power of a graph G

$B^{\mathcal{V}}$ – vertex B-matrix of a graph

$B^{\mathcal{E}}$ – edge B-matrix of a graph

$n_{\triangle}(G)$ – number of triangles in the graph G

Abbreviations

API – Application Programmer Interface

APSP – All-Pair Shortest-Paths

BFS – Breadth-First Search

CSV – Comma Separated Values

CUDA – Compute Unified Device Architecture

d.p. – definite positive graph kernel

GCC – Greatest Connected Component

Gin – *Graph Investigator* application

GPGPU – General-purpose computing on graphics processing units

GPU – Graphical Processing Unit

JNI – Java Native Interface

JRE – Java Runtime Environment

kPCA – kernel Principal Component Analysis

LDA – Linear Discriminant Analysis

LLE – Locally Linear Embedding

LPMIP – Locality-Preserved Maximum Information Projection

MM – Matrix Multiplication

MMC – Maximum Margin Criterion

PCA – Principal Component Analysis

SIMD – Single Instruction Multiple Data

SM – Streaming Multiprocessor

SP – Single Precision

SVD – Singluar Value Decomposition

Publications List

Publications related to the subject of the dissertation

1. **Wojciech Czech**, *Invariants of Distance k-Graphs for Graph Embedding*, accepted for publication in Pattern Recognition Letters, 2012.

2. **Wojciech Czech**, Witold Dzwinel, *Review of graph invariants for quantitative analysis of structure dynamics*, accepted for publication in the book: Advances in Intelligent Modeling and Simulation: Simulation Tools and Applications, eds. A. Byrski, Z. Oplatkova, M. Carvahlo, M. Kisiel-Dorohinicki, Springer, 2012.

3. **Wojciech Czech**, David A. Yuen, *Efficient graph comparison and visualization using GPU*, Proceedings of the 14th IEEE International Conference on Computational Science and Engineering (CSE 2011), IEEE Computer Society, 561-566, doi: 10.1109/CSE.2011.223, 2011.

4. **Wojciech Czech**, S. Goryczka, T. Arodz, W. Dzwinel, A. Dudek, *Exploring complex networks with Graph Investigator research application*, Computing and Informatics, vol. 30, no. 2, p. 381-410, 2011.

5. **Wojciech Czech**, *Graph Descriptors from B-Matrix Representation*, Graph-Based Representations in Pattern Recognition, Proc. of 8th IAPR-TC-15 International Workshop, GbRPR 2011, Lecture Notes in Computer Science, vol. 6658, 12-21, 2011.

6. R. Wcislo, W. Dzwinel, P. Gosztyla, D. A. Yuen, **W. Czech**, *Interactive Visualization Tool for Planning Cancer Treatment*, University of Minnesota Supercomputing Institute Research Report UMSI 2011/7, January 2011 and CB number 2011-4.

7. **Wojciech Czech**, *Methods for graph feature extraction in satellite photo recognition* (in Polish), Proceedings of Electrotechnical Institute, ISSN 0032-6216, no. 243, 2009.

8. **Wojciech Czech**, *Clustering of real-world data using multiple-graph representation and centrality measures*, Computational Intelligence: methods and applications, eds. Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, Jacek Zurada, ISBN 978-83-60434-50-5, 9th Conference on Artificial Intelligence and Soft Computing, 2008.

9. **Wojciech Czech**, *The methods for graph isomorphism testing* (in Polish), Zeszyty Studenckiego Towarzystwa Naukowego, ISSN 1732-0925, no. 14, 2008.

10. **Wojciech Czech**, *Determining graph isomorphism with topological descriptors* (in Polish, book chapter), Software engineering - theory and applications, eds. Zbigniew Huzar,

Zygmunt Mazur, Transport and Communication Publishers, ISBN 978-83-206-1703-0, 2008.

11. **Wojciech Czech**, *Application of algebraic graph descriptors for clustering of real-world structures*, Photonics applications in astronomy, communications, industry, and high-energy physics experiments VII, Proceedings of SPIE, vol. 6937, 2007.

Other publications

1. Yichen Zhou, Robin M. Weiss, Elizabeth McArthur, David Sanchez, Xiang Yao, Dave Yuen, Mike R. Knox, **W. Czech**, *WebViz: A Web-Based Collaborative Interactive Visualization System for Large-Scale Data Sets*, University of Minnesota Supercomputing Institute Research Report UMSI 2011/11, February 2011.

2. Jonathan C. McLane, **Wojciech Czech**, David A. Yuen, Mike R. Knox, Shuo Wang, Jim B. S. Greensky, Erik O. D. Sevre, *Ubiquitous interactive visualization of large-scale simulations in geosciences over a Java-based web-portal*, Concurrency and Computation: Practice and Experience, ISSN 1532-0626, vol. 22, 2010.

3. James B. S. G. Greensky, **Wojciech Czech**, David A. Yuen, Michael Richard Knox, Megan Rose Damon, Shi Steve Chen, M. Charley Kameyama, *Ubiquitous interactive visualization of 3D mantle convection using a web-portal with Java and Ajax framework*, Visual Geosciences, ISSN 1610-2924, vol. 13, no. 1, 2008.

4. J. C. McLane, **W. Czech**, D. A. Yuen, J. Greensky, M. R. Konx, *Interactive visualization of 3-D mantle convection extended through AJAX applications*, AGU 2008 Fall Meeting 15–19 December San Francisco, American Geophysical Union, IN23B-1094, 2008.

5. D. Bhattarai, **Wojciech Czech**, B.B. Karki, David A. Yuen, *Web-based implementation of atomistic visualization*, AGU 2008 Fall Meeting 15–19 December San Francisco, American Geophysical Union, MR21B-1788, 2008.

6. Jonathan C. McLane, **Wojciech Czech**, David A. Yuen, Michael R. Knox, James B. S. G. Greensky, M. Charley Kameyama, Vincent M. Wheeler, Rahul Panday, Hiroki Senshu, *Ubiquitous interactive visualization of 3-D mantle convection through Web applications using Java*, Advances in visual computing: 4th international symposium, ISVC 2008: Las Vegas, Lecture Notes in Computer Science, ISSN 0302-9743, 2008.

Computing and Informatics, Vol. 30, 2011, 1001–1031, V 2011-Feb-9

EXPLORING COMPLEX NETWORKS WITH GRAPH INVESTIGATOR RESEARCH APPLICATION

Wojciech Czech, Witold Dzwinel

Institute of Computer Science
AGH University of Science and Technology
Al. Mickiewicza 30
30-059 Kraków, Poland
e-mail: czech@agh.edu.pl

Sławomir Goryczka

Department of Mathematics and Computer Science
Emory University
400 Dowman Drive, Atlanta, GA 30322, USA

Tomasz Arodź

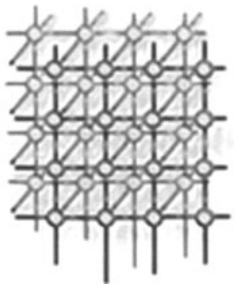
Department of Computer Science
School of Engineering, Virginia Commonwealth University
401 West Main Street, Richmond, Virginia 23284-3019, USA

Arkadiusz Z. Dudek

Oncology and Transplantation (HOT) Division
Medical School, University of Minnesota
420 Delaware Street S. E., Minneapolis, MN 55455, USA

Manuscript received 9 November 2009; revised 18 January 2011
Communicated by Jacek Kitowski

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2009)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1532



Ubiquitous interactive visualization of large-scale simulations in geosciences over a Java-based web-portal

Jonathan C. McLane[1,*,†], W. Walter Czech[2], David A. Yuen[1], Mike R. Knox[3], Shuo Wang[1], Jim B. S. Greensky[3] and Erik O. D. Sevre[1]

[1]*Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455, U.S.A.*
[2]*Computer Science Institute, AGH University, Krakow, Poland*
[3]*Laboratory of Computational Science and Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A.*

SUMMARY

We have designed a web-based software system for real-time interactive visualization of results taken from large-scale simulations of 3-D mantle convection and other large-scale modeling efforts. This approach allows for intense visualization sessions for a couple of hours as opposed to storing massive amounts of data in a storage system. Our data sets consist of 3-D data with over 10 million unknowns at each time-step to be used for volume rendering. Large-scale interactive visualization on a display wall holding 15 million pixels has already been accomplished. With the advent of the age of the mobile web, we have now extended this tactic to hand-held devices, such as the OQO, Nokia N800, netbooks, iPHONE, and Android OS phones. We are developing web-based software in Java to extend the ubiquitous use of this system across long distances. The software is aimed at creating an interactive and functional application capable of running on multiple browsers by taking advantage of two AJAX-enabled web frameworks: Echo2 (http://echo.nextapp.com/site/echo2) and Google Web Toolkit (http://code.google.com/webtoolkit/). Modular building of the system allows for components to be swapped out so that other forms of visualization can be accommodated, such as molecular dynamics in mineral physics or 2-D data sets from the regional convection models. Copyright © 2009 John Wiley & Sons, Ltd.

Received 15 January 2009; Accepted 24 July 2009

KEY WORDS: interactive visualization; 3-D mantle convection; Java

*Correspondence to: Jonathan C. McLane, Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455, U.S.A.
†E-mail: gordon.diehard@gmail.com

Contract/grant sponsor: National Science Foundation

Invariants of Distance k-Graphs for Graph Embedding

Wojciech Czech

AGH University of Science and Technology,
Institute of Computer Science,
Al. Mickiewicza 30, 30-059 Kraków, Poland
czech@agh.edu.pl

Abstract

Graph comparison algorithms based on metric space embedding have been proven robust in graph clustering and classification. In this paper we propose graph embedding method exploiting ordered invariants of distance k-graphs, which encode structure of shortest-paths. We study degree histograms of those graphs and use them to construct permutation invariant graph representation called vertex B-matrix. In order to extract more information from structural patterns we also define edge distance k-graphs and associated edge B-matrix. Next, several new graph characteristics obtained by condensing information stored in B-matrices are introduced. We demonstrate that our approach provides stable embedding, which capture relevant graph features. Experiments on classification with satellite photo and mutagenicity benchmark datasets revealed, that new descriptors allow for distinguishing graphs with non trivial structural differences. Moreover, they appear to outperform descriptors based on heat kernel matrix, being at the same time more effective computationally. In the end we test feature selection on B-matrices showing that selecting right B-submatrix can improve classification rate on testing datasets.

Keywords: graph embedding, graph invariants, b-matrix

1. Introduction

With a continuous rise of structured data volume, graph mining becomes complex and computationally challenging task. This is observed both in structural pattern recognition and complex networks areas, where increasing

List of Figures

List of Tables

Index

Bibliography

[1] http://cs.anu.edu.au/~bdm/nauty/.

[2] http://www.greyc.ensicaen.fr/iapr-tc15/.

[3] http://nd.edu/~networks/resources.htm.

[4] http://evolution.genetics.washington.edu/phylip.html.

[5] https://networkx.lanl.gov/wiki.

[6] http://www.netminer.com.

[7] http://www.orgnet.com.

[8] http://jung.sourceforge.net.

[9] http://prefuse.org.

[10] http://www.amiravis.com.

[11] http://snap.stanford.edu/data/.

[12] Cuda c best practices guide, March 2011.

[13] Cuda c programming guide, June 2011.

[14] V. Ágoston, P. Csermely, and S. Pongor. Multiple weak hits confuse complex systems: a transcriptional regulatory network as an example. *Physical Review E*, 71(5):51909, 2005.

[15] R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[16] M. Arita. The metabolic world of Escherichia coli is not small. *Proceedings of the National Academy of Sciences of the United States of America*, 101(6):1543, 2004.

[17] T. Arodz. Clustering organisms using metabolic networks. In *Proceedings of the 8th international conference on Computational Science, Part II*, ICCS '08, pages 527–534, Berlin, Heidelberg, 2008. Springer-Verlag.

[18] T. Arodz. Conservation of Edge Essentiality Profiles in Metabolic Networks Across Species. *Complex Sciences*, pages 1865–1876, 2009.

[19] L. Babai and E.M. Luks. Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 171–183. ACM, 1983.

[20] JP Bagrow, EM Bollt, and JD Skufca. Portraits of complex networks. *EPL (Europhysics Letters)*, 81:68004, 2008.

[21] X. Bai, R. Wilson, and E. Hancock. Manifold embedding of graphs using the heat kernel. *Mathematics of Surfaces XI*, pages 34–49, 2005.

[22] J.W. Baish and R.K. Jain. Fractals and cancer. *Cancer research*, 60(14):3683, 2000.

[23] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509, 1999.

[24] A.L. Barabási and Z.N. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[25] A. Barrat, M. Barthlemy, and A. Vespignani. *Dynamical processes on complex networks*. Cambridge University Press, 2008.

[26] D.S. Bassett and E. Bullmore. Small-world brain networks. *The Neuroscientist*, 12(6):512, 2006.

[27] V. Batagelj, A. Mrvar, M. Juenger, and P. Mutzel. Analysis and visualization of large networks. *Graph drawing software*, pages 77–103, 2003.

[28] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.

[29] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, 2008.

[30] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[31] D. Bonchev and D.H. Rouvray. *Complexity in chemistry, biology, and ecology*. Springer Verlag, 2005.

[32] B. Bonev, F. Escolano, D. Giorgi, and S. Biasotti. High-dimensional spectral feature selection for 3d object recognition based on reeb graphs. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 119–128, 2010.

[33] K.M. Borgwardt. *Graph kernels [Ph. D. thesis]*. PhD thesis, 2007.

[34] K.L. Boyer and S. Sarkar. Perceptual organization in computer vision: status, challenges, and potential. *Computer Vision and Image Understanding*, 76(1):1–5, 1999.

[35] U. Brandes and C. Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, 17(7):2303, 2007.

[36] U. Brandes and D. Wagner. Analysis and visualization of social networks. *Accessed July*, 15:2008, 2008.

[37] A.E. Brouwer, A.M. Cohen, and A. Neumaier. *Distance-regular graphs*, volume 24. Springer-Verlag Berlin, 1989.

[38] A. Buluç, J.R. Gilbert, and C. Budak. Solving path problems on the gpu. *Parallel Computing*, 36(5-6):241–253, 2010.

[39] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[40] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, 1998.

[41] A. Caflisch. Network and graph analyses of folding free energy surfaces. *Current opinion in structural biology*, 16(1):71–78, 2006.

[42] P. Carmeliet and R.K. Jain. Angiogenesis in cancer and other diseases. *NATURE-LONDON*, pages 249–257, 2000.

[43] F. Chung and S.T. Yau. Discrete green's functions. *Journal of Combinatorial Theory, Series A*, 91(1-2):191–214, 2000.

[44] F.R.K. Chung. Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92). *American Mathematical Society*, 3:8, 1997.

[45] A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[46] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[47] D.J. Cook and L.B. Holder. *Mining graph data*. Wiley-Blackwell, 2007.

[48] Y. Cooper. Properties determined by the ihara zeta function of a graph. *The electronic journal of combinatorics*, 16(R84):1, 2009.

[49] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1367–1372, 2004.

[50] L.F. Costa, F.A. Rodrigues, G. Travieso, and P.R.V. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.

[51] P. Csermely, V. Ágoston, and S. Pongor. The efficiency of multi-target drugs: the network approach might help drug design. *Trends in pharmacological sciences*, 26(4):178–182, 2005.

[52] W. Czech. Application of algebraic graph descriptors for clustering of real-world structures. In *Photonics applications in astronomy, communications, industry, and high-energy physics experiments VII*, volume 6937 of *Proceedings of SPIE*, 2007.

[53] W. Czech. Clustering of real-world data using multiple-graph representation and centrality measures. In Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek Zurada, editors, *Computational Intelligence: methods and applications*, Proceedings of 9th Conference on Artificial Intelligence and Soft Computing, 2008.

[54] W. Czech. *Determining graph isomorphism with topological descriptors.* Software engineering - theory and applications. Transport and Communication Publishers, 2008. (in Polish).

[55] W. Czech. The methods for graph isomorphism testing. *Zeszyty Studenckiego Towarzystwa Naukowego*, (14), 2008. (in Polish).

[56] W. Czech. Methods for graph feature extraction in satellite photo recognition. *Proceedings of Electrotechnical Institute*, (243), 2009. (in Polish).

[57] W. Czech. Graph descriptors from b-matrix representation. In *Graph-Based Representations in Pattern Recognition, Proceedings of GbRPR 2011*, volume 6658 of *LNCS*, pages 12–21. Springer, 2011.

[58] W. Czech, S. Goryczka, T. Arodz, W. Dzwinel, and A. Dudek. Exploring complex networks with graph investigator research application. *Computing and Informatics*, 30(2), 2011.

[59] W. Czech and D. A. Yuen. Efficient graph comparison and visualization using gpu. *Proceedings of the 14th IEEE International Conference on Computational Science and Engineering (CSE 2011)*, pages 561–566, 2011.

[60] P. D'Alberto and A. Nicolau. R-kleene: A high-performance divide-and-conquer algorithm for the all-pair shortest path for densely connected networks. *Algorithmica*, 47(2):203–213, 2007.

[61] D. Eppstein and J. Wang. A steady state model for graph power laws. *Arxiv preprint cs/0204001*, 2002.

[62] P. Erdös and A. Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[63] E. Estrada. Generalized walks-based centrality measures for complex biological networks. *Journal of theoretical biology*, 263(4):556–565, 2010.

[64] E. Estrada, M. Fox, D.J. Higham, and GL Oppo. *Network Science. Complexity in Nature and Technology.* Springer, 2010.

[65] E. Estrada and D.J. Higham. Network properties revealed through matrix functions. *SIAM review*, 52(4):696–714, 2010.

[66] E. Estrada, D.J. Higham, and N. Hatano. Communicability betweenness in complex networks. *Physica A: Statistical Mechanics and its Applications*, 388(5):764–774, 2009.

[67] P. Foggia, C. Sansone, and M. Vento. A database of graphs for isomorphism and subgraph isomorphism benchmarking. *Jolion et al.[9]*, pages 176–187.

[68] J. Forman, P. Clemons, S. Schreiber, and S. Haggarty. Spectralnet–an application for spectral graph analysis and visualization. *BMC bioinformatics*, 6(1):260, 2005.

[69] S. Fortin. The graph isomorphism problem. *Department of Computing Science, University of Alberta*, 1996.

[70] C. Fouard, E. Cassot, G. Malandain, C. Mazel, S. Prohaska, D. Asselot, M. Westerhoff, and J.P. Marc-Vergnes. Skeletonization by blocks for large 3d datasets: application to brain microcirculation. In *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 89–92. IEEE, 2004.

[71] L.C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[72] L.C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[73] Thomas Gaertner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, pages 129–143. Springer-Verlag, August 2003.

[74] Y. Gazit, D.A. Berk, M. Leunig, L.T. Baxter, and R.K. Jain. Scale-invariant behavior and vascular network formation in normal and tumor tissue. *Physical review letters*, 75(12):2428–2431, 1995.

[75] A.H. Gebremedhin, A. Tarafdar, F. Manne, and A. Pothen. New acyclic and star coloring algorithms with application to computing hessians. *SIAM Journal on Scientific Computing*, 29(3):1042–1072, 2008.

[76] D. Gfeller, P. De Los Rios, A. Caflisch, and F. Rao. Complex network analysis of free-energy landscapes. *Proceedings of the National Academy of Sciences*, 104(6):1817, 2007.

[77] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821, 2002.

[78] C.D. Godsil and G. Royle. *Algebraic graph theory*. Springer New York, 2001.

[79] M. Gordon and GR Scantlebury. Non-random polycondensation: Statistical theory of the substitution effect. *Transactions of the Faraday Society*, 60:604–621, 1964.

[80] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1100–1111, 2005.

[81] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[82] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Transactions on Graphics (TOG)*, 4(2):74–123, 1985.

[83] S. Gunter and H. Bunke. Validation indices for graph clustering. *Pattern Recognition Letters*, 24(8):1107–1113, 2003.

[84] I. Gutman and N. Trinajstić. Graph theory and molecular orbitals. *New Concepts II*, pages 49–93, 1973.

[85] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[86] P. Harish and P. Narayanan. Accelerating large graph algorithms on the gpu using cuda. *High Performance Computing–HiPC 2007*, pages 197–208, 2007.

[87] P. Harish, V. Vineet, and PJ Narayanan. Large graph algorithms for massively multithreaded architectures. *International Institute of Information Technology, Tech. Rep. IIIT/TR/2009/74*, 2009.

[88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[89] P. Holme, M. Huss, and H. Jeong. Subnetwork hierarchies of biochemical pathways. *Bioinformatics*, 19(4):532, 2003.

[90] M. Huisman and M.A.J. Van Duijn. Stocnet: Software for the statistical analysis of social networks. *Connections*, 25(1):7–26, 2003.

[91] A. Jamakovic and S. Uhlig. On the relationships between topological measures in real-world networks. *Networks and Heterogeneous Media*, 3(2):345, 2008.

[92] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[93] S. Jouili and S. Tabbone. Graph embedding using constant shift embedding. *Recognizing Patterns in Signals, Speech, Images and Videos*, pages 83–92, 2010.

[94] M. Kaiser. Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks. *New Journal of Physics*, 10:083042, 2008.

[95] G. Kalna and D.J. Higham. A clustering coefficient for weighted networks, with application to gene expression data. *Ai Communications*, 20(4):263–271, 2007.

[96] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pages 321–328, 2003.

[97] F. Képès. *Biological networks*. World Scientific Pub Co Inc, 2007.

[98] L.B. Kier and L.H. Hall. *Molecular connectivity in structure-activity analysis*. Research Studies Press: Letchworth, Hertfordshire, England, 1986.

[99] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002.

[100] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[101] J.M. Kleinberg. Small-world phenomena and the dynamics of information. *Science*, 14(12):1–14, 2001.

[102] J. Köbler, U. Schöning, and J. Torán. *The graph isomorphism problem: its structural complexity*. Springer, 1993.

[103] B. Le Saux and H. Bunke. Feature selection for graph-based image classifiers. *Pattern Recognition and Image Analysis*, pages 147–154, 2005.

[104] V.W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A.D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 451–460. ACM, 2010.

[105] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[106] H. Li, T. Jiang, and K. Zhang. Efficient and robust feature extraction by maximum margin criterion. *Neural Networks, IEEE Transactions on*, 17(1):157–165, 2006.

[107] B. Luo, R.C. Wilson, and E.R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213–2230, 2003.

[108] Y.P. Luo, H.Y. Lin, M.C. Huang, and T.M. Liaw. Conformation-networks of two-dimensional lattice homopolymers. *Physics Letters A*, 359(3):211–217, 2006.

[109] H.W. Ma and A.P. Zeng. The connectivity structure, giant strong component and centrality of metabolic networks. *Bioinformatics*, 19(11):1423, 2003.

[110] D.J. Marchette. *Random graphs for statistical pattern recognition*. Wiley-IEEE, 2004.

[111] R. Marfil, F. Escolano, and A. Bandera. Graph-based representations in pattern recognition and computational intelligence. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence*, IWANN '09, pages 399–406, Berlin, Heidelberg, 2009. Springer-Verlag.

[112] D.M. McDonald and P. Baluk. Imaging of angiogenesis in inflamed airways and tumors: Newly formed blood vessels are not alike and may be wildly abnormal. *Chest*, 128(6 suppl):602S, 2005.

[113] B.D. McKay and Vanderbilt University. Dept. of Computer Science. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[114] S. Mika, B. Schölkopf, A.J. Smola, K.R. Müller, M. Scholz, and G. Rätsch. Kernel pca and de-noising in feature spaces. *Advances in neural information processing systems*, 11(1):536–542, 1999.

[115] S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (coil-20). *Dept. Comput. Sci., Columbia Univ., New York.[Online] http://www. cs. columbia. edu/CAVE/coil-20. html*, 1996.

[116] M. Neuhaus and H. Bunke. *Bridging the gap between graph edit distance and kernel machines*, volume 68. World Scientific Pub Co Inc, 2007.

[117] M.E.J. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[118] M.E.J. Newman. The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, 2007.

[119] S. Nikolic, G. Kovacevic, A. Milicevic, and N. Trinajstic. The zagreb indices 30 years after. *Croatica chemica acta*, 76(2):113–124, 2003.

[120] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[121] J.R. Platt. Influence of neighbor bonds on additive bond properties in paraffins. *The Journal of Chemical Physics*, 15:419, 1947.

[122] J. Podani, Z.N. Oltvai, H. Jeong, B. Tombor, A.L. Barabási, and E. Szathmary. Comparable system-level organization of Archaea and Eukaryotes. *Nature Genetics*, 29(1):54–56, 2001.

[123] H.J. Qiu and E.R. Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007.

[124] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[125] S.A. Rahman, P. Advani, R. Schunk, R. Schrader, and D. Schomburg. Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). *Bioinformatics*, 21(7):1189, 2005.

[126] L. Ralaivola, S.J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

[127] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, pages 846–850, 1971.

[128] M. Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23):6609–6615, 1975.

[129] F. Rao and A. Caflisch. The protein folding network. *Journal of molecular biology*, 342(1):299–306, 2004.

[130] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, and A.L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551, 2002.

[131] P. Ren, R.C. Wilson, and E.R. Hancock. Graph characterization via ihara coefficients. *Neural Networks, IEEE Transactions on*, 22(2):233–245, 2011.

[132] K. Riesen and H. Bunke. Iam graph database repository for graph based pattern recognition and machine learning. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, 2008.

[133] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.

[134] K. Riesen and H. Bunke. Graph classification based on vector space embedding. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(6):1053–1081, 2009.

[135] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *Arxiv preprint cs/0209028*, 2002.

[136] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

[137] G. Rücker and C. Rücker. Walk counts, labyrinthicity, and complexity of acyclic and cyclic graphs and molecules. *Journal of Chemical Information and Computer Sciences*, 40(1):99–106, 2000.

[138] A. Sanfeliu and Fu King-Sun. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 13(3):353–362, 1983.

[139] K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, and S.W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

[140] S. Skiena. *Implementing discrete mathematics: combinatorics and graph theory with Mathematica*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[141] C. Song, S. Havlin, and H.A. Makse. Self-similarity of complex networks. *Nature*, 433, 2005.

[142] Daniel A. Spielman. Spectral graph theory and its applications - course notes. http://www.cs.yale.edu/homes/spielman/eigs/.

[143] F. Suard, V. Guigue, A. Rakotomamonjy, and A. Benshrair. Pedestrian detection using stereo-vision and graph kernels. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 267–272. IEEE, 2005.

[144] K. Supekar, V. Menon, D. Rubin, M. Musen, and M.D. Greicius. Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS computational biology*, 4(6):e1000100, 2008.

[145] R. Tanaka. Scale-rich metabolic networks. *Physical review letters*, 94(16):168101, 2005.

[146] Y. Tao and W.I. Grosky. Delaunay triangulation for image object indexing: A novel method for shape representation. In *Proc. 7th SPIE Symposium on Storage and Retrieval for Image and Video Databases*, pages 631–642. Citeseer, 1999.

[147] J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

[148] P. Topa. Dynamically reorganising vascular networks modelled using cellular automata approach. *Cellular Automata*, pages 494–499, 2010.

[149] Z. Toroczkai and K.E. Bassler. Jamming is limited in scale-free systems. *Nature-London*, pages 716–716, 2004.

[150] W.T. Tutte. How to draw a graph. *Proc. London Math. Soc*, 13(3):743–768, 1963.

[151] M. Tyomkyn and A. Uzzell. Distances in graphs. *Arxiv preprint arXiv: 1011.2450*, 2010.

[152] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[153] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[154] V. Volkov and J.W. Demmel. Benchmarking gpus to tune dense linear algebra. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–11. IEEE, 2008.

[155] A. Wagner and D.A. Fell. The small world inside large metabolic networks. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1478):1803, 2001.

[156] H. Wang, S. Chen, Z. Hu, and W. Zheng. Locality-preserved maximum information projection. *Neural Networks, IEEE Transactions on*, 19(4):571–585, 2008.

[157] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge Univ Pr, 1994.

[158] D.J. Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton Univ Pr, 2003.

[159] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442, 1998.

[160] R. Wcisło, W. Dzwinel, P. Gosztyla, D. A. Yuen, and W. Czech. Interactive visualization tool for planning cancer treatment. University of Minnesota Supercomputing Institute Research Report UMSI 2011/7, CB number 2011-4, January 2011.

[161] R. Wcisło, W. Dzwinel, D.A. Yuen, and A.Z. Dudek. A 3-d model of tumor progression based on complex automata driven by particle dynamics. *Journal of Molecular Modeling*, 15(12):1517–1539, 2009.

[162] M. Welter and H. Rieger. Physical determinants of vascular network remodeling during tumor growth. *Eur. Phys. J. E*, 33:149–163, 2010.

[163] H. Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1):17–20, 1947.

[164] R.C. Wilson, E.R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1112–1124, 2005.

[165] B. Xiao, E.R. Hancock, and R.C. Wilson. A generative model for graph matching and embedding. *Computer Vision and Image Understanding*, 113(7):777–789, 2009.

[166] B. Xiao, E.R. Hancock, and R.C. Wilson. Graph characteristics from the heat kernel trace. *Pattern Recognition*, 42(11):2589–2606, 2009.

[167] M.A. Yildirim, K.I. Goh, M.E. Cusick, A.L. Barabási, and M. Vidal. Drug-target network. *Nature biotechnology*, 25(10):1119, 2007.

[168] B. Zhang and S. Horvath. A general framework for weighted gene co-expression network analysis. *Statistical applications in genetics and molecular biology*, 4(1):17, 2005.

[169] D. Zhu and Z.S. Qin. Structural comparison of metabolic networks in selected single cell organisms. *BMC Bioinformatics*, 6(1):8, 2005.