View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarWorks @ Georgia State University

Georgia State University
ScholarWorks @ Georgia State University

Computer Science Dissertations

Department of Computer Science

12-2009

Metabolic Network Alignments and their Applications

Qiong Cheng
Georgia State University

Follow this and additional works at: https://scholarworks.gsu.edu/cs_diss

Part of the Computer Sciences Commons

Recommended Citation
Cheng, Qiong, "Metabolic Network Alignments and their Applications." Dissertation, Georgia State University, 2009.
https://scholarworks.gsu.edu/cs_diss/44

This Dissertation is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

METABOLIC NETWORK ALIGNMENTS AND THEIR APPLICATIONS

by

QIONG CHENG

Under the direction of Alexander Zelikovsky

ABSTRACT

The accumulation of high-throughput genomic and proteomic data allows for the reconstruction of the increasingly large and complex metabolic networks. In order to analyze the accumulated data and reconstructed networks, it is critical to identify network patterns and evolutionary relations between metabolic networks. But even finding similar networks becomes computationally challenging. The dissertation addresses these challenges with discrete optimization and the corresponding algorithmic techniques.

Based on the property of the gene duplication and function sharing in biological network, we have formulated the network alignment problem which asks the optimal vertex-to-vertex mapping allowing path contraction, vertex deletion, and vertex insertions. We have proposed the first polynomial time algorithm for aligning an acyclic metabolic pattern pathway with an arbitrary metabolic network. We also have proposed a polynomial-time algorithm for patterns with small treewidth and implemented it for series-parallel patterns which are commonly found among metabolic networks. We have developed the metabolic network alignment tool for free public use.

We have performed pairwise mapping of all pathways among five organisms and found a set of statistically significant pathway similarities. We also have applied the network alignment to identifying inconsistency, inferring missing enzymes , and finding potential candidates.

INDEX WORDS: Network alignment, Graph pattern matching

METABOLIC NETWORK ALIGNMENTS AND THEIR APPLICATIONS

by

QIONG CHENG

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2009

Copyright by
Qiong Cheng
2009

METABOLIC NETWORK ALIGNMENTS AND THEIR APPLICATIONS

by

QIONG CHENG

Committee Chair: Alexander Zelikovsky

Committee: John Houghton
Xiaolin Hu
Sushil K. Prasad

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
December 2009

To Mom, Dad, Jinpeng and Ellen for love and support.

ACKNOWLEDGMENTS

I owe much to my advisor, Dr. Alexander Zelikovsky. I have learned a lot about computer science and conducting research from him. Thank him for teaching me. Without his guidance, the project doesn't exist. Many thanks to my committee members: Dr. John Houghton, Dr. Xiaolin Hu, Dr. Sushil K. Prasad. Their comments on my work have helped a lot. Also thanks to Dr. Robert Harrison and Dr. Piotr Berman(in PSU), who have injected their knowledge to the research.

A special thank goes out to my good friends Gulsah, Dima, Irina, Jim, Weidong, Stefan, and Serghei. We all shared experience of graduate school together that bonded us and we will remember it for the remainder of our lives.

More than to anyone else, I owe to the love and support of my family: My mother Lieyu Liu, my father Tianzhong Cheng, my sister Lin Cheng, my husband Jinpeng Wei, my daughter Ellen. Despite all my research work, you are the best part of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Metabolism is a vital cellular process whose understanding is critical to human disease studies and drug discovery. The accumulation of high-throughput genomic, proteomic and metabolical data allows for increasingly accurate modeling and reconstruction of metabolic networks for a variety of organisms such as Escherichia coli, Saccharomyces cerevisiae and humans.

The wealth of the network information have been stored in several public collections of databases, including KEGG ([31]) and BioCyc ([2]). BioCyc ([2]) provides electronic reference sources on the pathways and genomes of different organisms. KEGG ([31]) is a collection of manually wiring diagrams of molecular interactions, reactions, and relations. The collection represents authors' knowledge on the molecular interaction and reaction networks for metabolism, genetic information processing, environmental information processing, cellular processes, human diseases, and drug development. These databases provide tools for pathway visualization and for queries on pathway components such as substrates, products and reactions. However, computational tools are called for transferring the well studied knowledge in databases to unknown one (e.g. for searching for homologues to a query pathway in a collection of known pathways) and for aligning two pathways to locate conserved pathway fragments. The need will increase over the next several years as biologists begin to inspect the existing pathways but also to redirect and re-engineer metabolic pathways.

For example, biologists working on chronic diseases such as breast cancer and obesity spend years have constructed a small portion of a pathway by analyzing and comparing experimental data. The entire process takes many years. The time and energy consuming (non-trivial) task can be expressed as a series of the pathway queries. The query can reveal clues about what information might be missing or spurious in the constructed query graph.

Additionally, most of the high-throughput genomic, proteomic data are not catalogued or processed directly by hand but by computational technologies. It means that the high-throughput datasets may be filled with technical and biological noise and have different technical biases and coverage ([29]) or the usage of inconsistent data/tool versions. All of them requires data curation. However not all data are curated. For BioCyc, only a part of pathway data have received person-decades of literature-based curation and are the most accurate ([2]). However the logical interpretation of the whole network is definitely not easily comprehensible to the human brain and therefore it is impossible to curate all the more and more increasing data only by hand. The computational tool is required for inferring the missing/inconsistent information in genome and pathway databases.

For example, we observed the glutamate degradation VII pathway in *B. subtilis* and found out that there exist two partially identified enzymes (1.2.4.- and 2.3.1.-). The partially identified enzymes mean that no proper gene or its product has the right function in an organism when the pathway was constructed. But with the growth of genomic and proteomic database, the function has been assigned to some gene or its product but the pathway has not been curated yet. Inspecting the existing pathways (by searching for homologues to a query pathway in a collection of known pathways across different species or in the same species) can help resolve the ambiguity of databases.

Furthermore, comparison of metabolic pathways in a healthy and diseased cells can facilitate the research of disease association and drug discovery because it can be used to study the evolution of the specified disease and the disruption of pathways in diseased cells.

Let the *pattern* be a pathway for which one is searching for homologous pathways in the *text*, i.e., the known metabolic network of a different species. Alignment of two networks, pattern and text, is a basic task which can meet the requirement of a series of open questions such as network evolution and critical target search. This is a challenging research topic from both biological and computational perspective.

In order to comprehensively search and mine metabolic pathways we formulated the problem as optimal network alignment and developed MetNetAligner, a novel web service tool for network comparison and query that are based on a powerful and efficient optimization algorithm for aligning labeled graphs.

We extracted the metabolic pathway data and modeled them as directed graphs. Furthermore, we employed MetNetAligner to conduct a study on the similarities and variations in the metabolic networks, which take into account the similarity of network topology and the enzymes' functions. We designed a novel scoring scheme and two fast dynamic programming based algorithms for metabolic network alignment, which applied feedback vertex set and tree decomposition techniques respectively. The algorithms find an optimal alignment between arbitrary pattern and text graphs allowing enzyme deletion and insertion as well as matching similar enzymes. The feedback vertex set techniques based algorithm is efficient for aligning pattern graphs with a restricted cyclic structure to arbitrary text graphs; the tree decomposition techniques based algorithm is efficient for arbitrary text graphs and pattern graphs with a bounded treewidth. We have implemented a first efficient algorithm finding an optimal alignment from a series-parallel pattern to an arbitrary text graph. Due to mimicking evolutionary machinery of gene duplication ([48]), similarly to [61], our algorithm allows to map different pattern enzymes into the same text enzyme.

MetNetAligner provides simple and intuitive web-interfaces and several services such as pathway retrieval, extraction, visualization and upload services. It can be used for predicting unknown pathways, comparing and finding conserved patterns, and resolving ambiguous identification of enzymes.

Further, we applied our method and tool to aligning metabolic pathways among five organisms (E. coli, S. cerevisiae, B. subtilis, Halobacterium sp. NRC-1, and T. thermophilus) and found a reasonably large set of statistically significant motifs which are not caused by database curating. We observed advantages of allowing pattern vertex deletions and the biological relevance of the pathway alignments.

We also extended our alignment tool to identify and fill pathway holes. A metabolic pathway hole may be the result of ambiguity, inconsistency, or mistakes in databases ([28]). With the further research, some of those proteins should get specific annotations and pathway descriptions should be updated in pathway/genome databases. As a consequence, filling pathway holes can facilitate the improvement of both the completeness and accuracy of the pathway database and the annotation of its associated genome.

The remainder of this proposal is organized as follows: Chapter 2 describes the model of metabolic systems. Chapter 3 presents the previous and related work. Chapter 4 presents the optimal network alignment problem: graph theoretical model of pathways, existing approaches to network mapping, as well as our formulation. Chapter 5 presents network alignment from a polytree to arbitrary graph, beginning with the definition and notations, solutions, and statistical significance and finishing with experimental results. Chapter 6 generalizes the optimal network alignment in arbitrary pattern graphs. Chapter 7 describes our MetNetAligner tool including implementation and features, manual, and validation and discussion. Chapter 8 describes an application of our network alignment algorithm and tool.

Publications related to the dissertation:

- Q. Cheng, R. Harrison, and A. Zelikovsky. "MetNetAligner: a web service tool for metabolic network alignments". Bioinformatics. Advanced Access published May 4, 2009.

- Q. Cheng, P. Berman, R. Harrison and A. Zelikovsky, "Fast Alignments of Metabolic Networks ", Proc. of IEEE International conference on Bioinformatics and Biomedicine (BIBM 2008), pp 147-152

- Q. Cheng, D. Kaur, R. Harrison, and A. Zelikovsky, "Mapping and Filling Metabolic Pathways ", RECOMB Satellite Conference on Systems Biology 2007

- Q. Cheng, R. Harrison, and A. Zelikovsky, ”Homomorphisms of Multisource Trees into Networks with Applications to Metabolic Pathways”, Proc. of IEEE 7-th International Symposium on BioInformatics and BioEngineering (BIBE’07)
- Q. Cheng, A. Zelikovsky, Network Mapping of Metabolic Pathways, Analysis of Complex Networks: From Biology to Linguistics, Wiley-VCH , ISBN 978-3-527-32345-6, pp 271-293

CHAPTER 2
MODELS OF METABOLIC SYSTEMS

Metabolic systems are vital cellular processes. In this chapter we first describe the background information and the concept of metabolic chemical reactions, metabolic pathways, and metabolic networks. Then we present the current data presentation and two databases which provide the collection of metabolic pathways. Finally we will discuss some open research topics, all of which need network alignments as a basic research.

2.1 Biological background

Metabolism is a set of vital cellular processes, each of which is composed of a set of chemical reactions. The reactions happen in the cells of living organisms. Their occurrences meet the requirement of the growth and reproduction of cell and even living organisms, maintain their structures and functions by consuming energy, and respond to their environments. It is a dynamic process. As a consequence, a cell is never at chemical equilibrium; to maintain life, it requires a constant generation and consumption of energy by the reactions and processes.

A disease is developed if its subnetwork or module that carries out the malignant function becomes robust and perpetual. Each of the subnetworks or modules is resulted from different causes in different individuals. As a consequence, individualized disease networks have in common some pathway components and the misdirected functions.

Drug studies for individual patients should consider irreversibly disrupting the robustness of a disease network. The irreversibility means that drugs will restore the original "normal" network, which is because there exist the general complexity and evolutionary nature of the development in the malignant network. Furthermore, drugs must redirect a malignant information flow to restore the normal and healthy pathway.

From the above discussion, we can draw a conclusion that the understanding, comparison, and analysis of the processes are critical to human disease studies and drug discovery. In the rest of this chapter, we will discuss the basic entities in metabolism.

2.2 Metabolic chemical reactions

The chemical reactions of metabolism are called metabolic chemical reactions. Numerous metabolic chemical reactions occurring in a cell are responsible for generating or consuming energy and maintaining the structure and function of the cell.

2.2.1 Enzymes

Almost all of the metabolic chemical reactions in organisms are performed by special functional proteins which are called **enzymes**. Enzymes as catalysts help the progress of chemical reactions so that the reactions reach their equilibrium. They allow these reactions to proceed quickly and efficiently and to respond the changes in the cell's environment or signals from other cells. Enzymes are highly specific and usually catalyze only a single reaction with exactly defined reactants.

2.2.2 Enzyme commission number

Enzymes are usually labeled with an Enzyme Commission number (EC number), which is expressed with a 4-level hierarchical scheme that classifies enzymes on a functional basis. The 4-digit EC number, $d_1.d_2.d_3.d_4$ represents a sub-sub-subclass indication of biochemical reaction. The top level is the first level which represents the reaction class (At this level, there are six broad classes of enzyme activity); the second level represents reaction subclass; the third level represents specific acceptor; the fourth level represents identifiers of reactions. In this way, an enzyme is classified by four numbers (e.g. 5.3.1.6), the first one representing the top level classification and the three following numbers the subsequent refinements thereof.

The more numbers the EC codes of two enzymes have in common from the top level to the bottom level, the more similar the reactions they catalyze are. For example, if $d1.d2$ of two enzymes are different, they are highly dissimilar; if $d1\&d2$ are same but $d3$ of two enzymes is different, they are somewhat similar; if $d1$, $d2, \&d3$ are same but $d4$ of two enzymes is different, they are relatively more similar. The property is known as a tight reaction property.

2.2.3 Chemical reactions

A chemical reaction is an event which occurs under certain conditions such as temperature or enzyme catalysts. It is a process that always involves chemical substances and leads to their interconversion. The initial substance or substances involved in the reaction are called reactants. The reactants on which enzymes act are also called substrates. The generated substances in the reaction are called as products. Chemical reactions are usually characterized by a chemical change. They consume one or more reactants and come up one or more products, which usually have properties different from the reactants (see 2.1). For the reaction which requires enzyme catalysis, an enzyme or enzymes can perform it. The change in the reaction reflects the forming and breaking of chemical bonds resulted from the motion of electrons. It also reflects the syntheses, combination, decompositions, or displacement among chemical substances. It determines the physiological and biochemical properties of a cell.

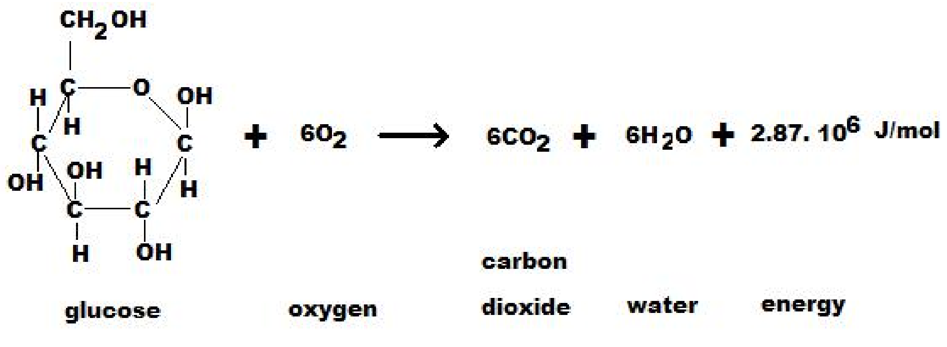


Figure 2.1. An example of chemical reaction.

2.2.4 Metabolites

The reactants, intermediates and products of these metabolic chemical reactions are referred to as metabolites. They are responsible for communication with the environment.A primary metabolite is directly involved in normal growth, development, and reproduction of the cell.

2.3 Metabolic pathways

Biochemical pathway is a building block of metabolism. It represents a flow of consecutive enzymatic reactions that take specific metabolites (also called substrate) as an input to yield specific products. In such pathways, a substrate is converted into a product by the first enzyme in the pathway and the product of the first reaction then becomes the substrate for the next reaction. The sequence of reactions continue until the final product is made. When a biochemical pathway is functioning, the initial substrate is continually converted to the final product through the series of steps in the pathway.

A metabolic pathway is the biochemical pathway that occurs within a cell and in which one chemical is transformed into another by a sequence of enzymes. Usually, metabolic pathways are divided into two classifications, namely catabolic pathways (also called degradation pathways) that break down high-energy compounds to release their free energy and anabolic pathways (also called biosynthetic pathways) that synthesize biomolecules from intermediate compounds, usually by using free energy. A classic example for a catabolic pathway is the oxidation of glucose sugar into water and carbon dioxide (see 2.1)([3]).

2.4 Metabolic networks

A metabolic network represents the system of connected chemical reactions. Metabolic networks are directed networks that represent the complete set of metabolic pathways. They can be observed to have typical topological behavior of scale free networks (power law distribution). They are robust, redundant, flexible, extremely controlled and follow nature laws of thermodynamics.

Take as a metabolic network example the subsystem for synthesis of the amino-acids valine, leucine and isoleucine in *Escherichia coli* (see 2.2). This image was taken from the Kyoto Encyclopedia of Genes and Genomes (KEGG) ([31]) database. The green boxes indicate enzymes which have been identified in the organism, in the case of Escherichia coli.

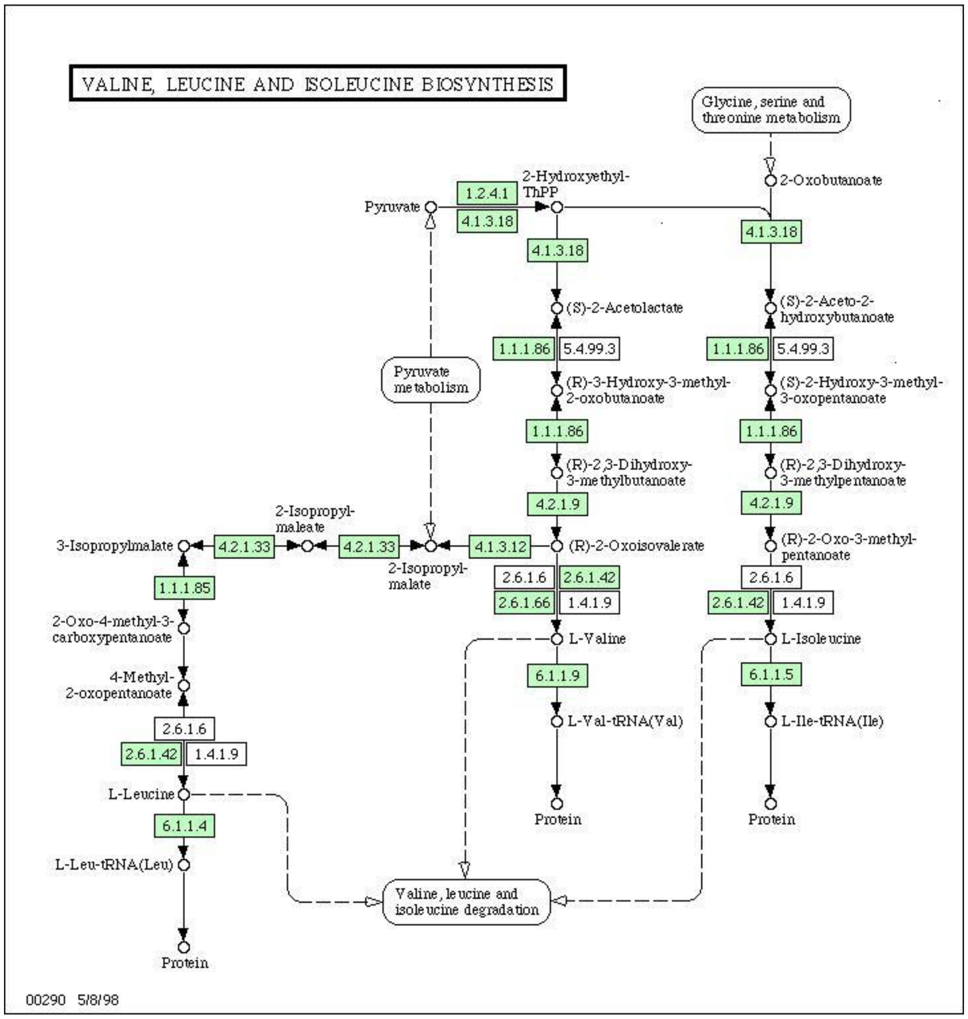


Figure 2.2. An example of metabolic network : the subsystem for synthesis of the amino-acids valine, leucine and isoleucine.

2.5 Metabolic network representation in databases

With the growth of high-throughput data collection, algorithms for network integration and comparison have been required and described. All of the work need to integrate multiple data types, incorporate explicit models of uncertainty, and include ontologically typed edges and nodes, which have driven us for using a popular network exchange file format.

BioPax, SBML and PSI-MI are the most superior. BioPax was originally developed for exchanging pathway data between databases such as KEGG and Ecocyc, SBML was designed for time dependent modeling, and PSI-MI was built for describing the results of high throughput experiments. The others such as SIF, XML, KGML have certain restrictions. The BioPax and SBML format are popular (They support the subcellular localization). There exist good tools such as KEGG2SBML, KEGG2BioPax, SBML2BioPax, which can be used to convert the file format from one type to another.

There exist two collections, BioCyc and KEGG PATHWAY, either of which supports different exchange file format (or model language). BioCyc supports BioPax. KEGG PATHWAY supports KGML. Due to the above available tools to convert the file formats, we choose the data collection by the characteristics of different applications and not by the file formats the collections supports.

2.5.1 BioCyc

BioCyc is a collection of 409 Pathway/Genome Databases. The BioCyc collection of Pathway/Genome Databases (PGDBs) provides electronic reference sources on the pathways and genomes of different organisms. Each database in the collection describes the genome and metabolic pathways of a single organism. The databases (DBs) are organized into tiers based on their quality ([2]).

Tier 1 databases have received person-decades of literature-based curation, and are the most accurate. Tier 2 and Tier 3 databases contain computationally predicted metabolic pathways with the predicted operons or genes code for missing enzymes in metabolic pathways ([2]).

2.5.2 KEGG PATHWAY

KEGG PATHWAY is a collection of manually wiring diagrams of molecular interactions, reactions, and relations. The collection represents authors' knowledge on the molecular interaction and reaction networks for metabolism, genetic information processing, environmental information processing, cellular processes, human diseases, and drug development ([3]).

2.6 Immediate Applications of Network Alignment

Let the pattern be a pathway for which we are searching for homologous pathways in the text, i.e., the known metabolic network of a different species or in the same species. Comparing pattern and text networks referred as network alignment becomes an important approach to explore biological networks that could provide insight into biological understanding and therapeutics. In this section we list several immediate applications of the network alignment approach.

2.6.1 Network similarity search

Network /group as a flexible structure has gained a broad array of applications such as predicting the structure of micro molecular, navigating complex metabolic networks, and analyzing biological network data. In the applications, there is a basic question: given two networks, find out how similarity they are. The similarity is dependent on applications. It is the need to be able to compare two networks and somehow assess their similarity. The similarity question has different forms such as: Are these graphs identical to each other? How many modification is needed to make two graphs identical? Can we find conserved regions in two or more such networks? If these graphs are evolved from the same ancestor, can we predict how far their evolution is?

To judge network similarity, we need to take into account not only the network properties including network labels but also network topology. Topology includes at least graph connectivity, degree distribution, clustering, diameter, and relative graphlet frequency distribution. The larger the number of the common properties is, the more likely it is that the two networks are similar.

Due to the complexity of network topology, the network similarity problem is non-trivial. Existing approaches are mostly based on isomorphic and homeomorphic embeddings, effectively solving a problem that is NP-complete [26] even when searching a match for a tree in acyclic networks. How to efficiently find out network similarity with the fewer topology limitations is still demanding.

2.6.2 Conserved network (or motif) search

In biology, conserved networks are similar or identical networks that occur within the others across species or in the same species.

Searching conserved pair-wise networks across species, for example, we can apply network alignments to find all species with nitrate reduction systems similar to that of Escherichia coli, or to examine the extent to which the cell division apparatus is conserved across a set of microbes. The conservation by comparing cellular networks of different species could enable evolutionary insights and it may indicate that a particular common structure/network may have been maintained by evolution.

Searching conserved pair-wise networks in the same species, for example, we can compare a diseased cellular network to a healthy one. The conservation may aid in finding a cure for the disease. Additionally, it may also indicate that the inconsistency happens in the databases.

Given a database of known networks, it is required in current real-world applications to identify the conserved motifs and to use the conserved ones to predict and validate interaction candidates.

The conserved network (or motif) search is a challenge problem from an algorithmic point of view. It includes determining the frequency of subgraph occurrences in a given graph, detecting isomorphy between these subgraphs, and detecting which subgraphs are overabundant in comparison to random graphs. The relation of conservation is symmetric but not transitive.

2.6.3 Discovering biological network evolution

Sequence comparison and alignment have had an enormous impact on our understanding of evolution, biology, and disease. Comparison and alignments of biological networks will likely have a similar impact.

The networks are allowed to not only grow but possibly wither over time or across species. Especially, the evolution of metabolic networks is characterized by gain as well as loss of reactions.

Network comparison can be used to identify the difference of pathways in different organisms or in the same organism. The graph edit distance can be used to measure the process of changes over time. Especially, for disease pathways, it can be used to discover the evolving process, which is useful for identifying critical targets.

Network evolution is challenging from both computation and biology point of view. The research aims at discovering and reflecting the dynamics of the system in a global view.

2.6.4 Predicting inconsistency and noise in database

With the growth of the number of genomic, proteomic and metabolical data, noise has been introduced into the system due to mistakes made during the recording of facts, due to the inconsistency among the different database versions, and due to errors in publications. For example, some proteins are predicted by bioinformatics tools, which may lead to incomplete information. Some biologically relevant pathways have been identified by the microarray experiments and corresponding tools, which possibly produces spurious gene expression data. The "Noise" may result from poor hybridizations, distortion during amplification, and the quality of clustering and function prediction.

Manual curation is the intuitive way to intend to either remove false or unreliable facts or to add missing links to a pathway. However, the logical interpretation of the whole network is not easily comprehensible to the human brain. As a result, the computational tool is required for inferring the missing/inconsistent information in genome and pathway databases.

Network alignment can be one of fundamentally computational approaches because the alignment can be used to transfer knowledge from a well-understood network to an unknown one. As a consequence, the inconsistency /noise in databases can be identified by the knowledge transfer. Biologists can save time and energy in the analysis and prediction.

2.6.5 Identifying hubs, clusters, or alternative pathways

Biological networks tend to have power law degree distribution (also called as scale-free network topology), which is characterized by hubs or clusters. Hubs are the vertices having the most connections in the networks. Hubs also can provide an idea about the information flows within the network, from major source hub targets to major sink hub targets [63]. Clusters are a subset of vertices that have higher link density between themselves as compared with the rest of the network [63]. Additionally, the immune system has its natural defense against infections by the alternative pathways of the complement systems. Disease can be described as broken pathways or as a disruption of a physical structure maintained by the network of interacting proteins. We can take the cancer development as an example.

The characteristics of network topology and the nature of the immune system lead to the robust network, in which a vertex or link can be removed without any influence in the health due to genetic mutations in evolution, somatic mutations occurring in a disease and throughout the life of the organism, and environmental conditions such as diet, trauma, and stress [63].

Identifying hubs, clusters, or alternative pathways is a difficult task to the research in disease association and human life. The process of identification involves the analysis and comparison of the network topologies.

CHAPTER 3
PREVIOUS AND RELATED WORK

The earlier discussion in network alignments was to formulate the problem as graph and subgraph isomorphism. An intuitive enumeration algorithm to obtain isomorphism of pattern graph P to text graph T, is to generate a state-space representation tree which represents all possible mappings between the nodes of the two graphs and to check whether each generated mapping in the tree is a good alignment. In the tree with vertex size of $\frac{|V_T|^{|V_P|+1}-1}{|V_T|-1}$, a vertex represents a pair of matched nodes; a path from a root down to a leaf represents a alignment between the two graphs. Any path across k levels in the tree represents a possible subgraph isomorphism between P and T.

For circumventing the hardness, a part of computation is filtered by using more selective feasibility rules to cut the state search space [25]. Another part is transferred into intensive preprocessing so that the alignment process based on subgraph isomorphism runs in polynomial time after the exponential preprocessing time is ignored. The first examples are Shasha et al [50], Yan et al [60] and X. Yan and J. Han [59], R. Giugno and D. Shasha [27] and Ferro et al [23], and B. T. Messmer [40] and Horst Bunke [14] which convert the database of graphs individually into DFS code trees, label-paths, and decisive trees.

The current discussion on this problem are mostly based on isomorphic and homeomorphic embeddings (see [53, 14, 40, 13, 25, 59, 50, 27, 5, 60, 21, 23, 35, 11] and [15, 16, 44, 43, 34, 33, 49, 39, 37, 55, 61, 19, 58] respectively) effectively solving a problem that is NP-complete [26] even when searching a match for a tree in acyclic networks.

The approaches to iso- and homeomorphism embedding restrict the size (see [49],[61]) or topology of the pattern (see [15],[16],[43],[19]) or use heuristics or approximation algorithms.

GraphMatch ([61]) allows to delete disassociated vertices or induced subnetwork in query network and then aligns its remainder to target networks by subgraph isomorphism.

In the following subsections, I first present some approaches proposed by Kelly et at.[49], Pinter et al.[43], and Yang and Sze[61], which is most related to our work. I also present some other approaches proposed by Koyuturk et al ([37]), Li et al ([39]), Sharon et al.[22, 12].

3.1 PathBlast

Kelly et al ([34, 33, 49]) has taken into account the nonlinearity of protein network topology and formulated the mapping problem as follows. Given a linear length-ℓ pathway *pattern* $T = (V_P, E_P)$ and *text* graph $G = (V_G, E_G)$, find an image of the pattern in the text without consecutive gaps and minimizing mismatches between proteins. A global alignment graph in [34] was built in which each vertex represents a pair of proteins and each edge represents a conserved interaction, gap, or mismatches; their objective is to find the k-highest-scoring path with limited length ℓ and no consecutive gaps or mismatches based on the built global graph. The approach takes $O(|V_T|^{\ell+2}|V_G|^2)$.

PathBlast is the web tool presented in [33]. They have developed with the same problem formulation as [34]. However, PathBlast's solution is to randomly decompose the text graph into linear pathways which are then aligned against the pattern and then to obtain optimal mapping based on standard sequence alignment algorithms. The algorithm requires $O(\ell!)$ random decompositions to ensure that no significant alignment is missed. Authors limited the size of the query to about six vertices.

3.2 MetaPathwayHunter: Pinter's approach and tool

Pinter et al ([43]) model metabolic pathways as outgoing trees; they reduce the problem to the approximately labeled tree homeomorphism problem. Given two undirected labeled trees P and T , and a scoring table that specifies the similarity scores between the label of any node appearing in T and the label of any node appearing in P, as well as a predefined node deletion (gap) penalty λ, to find a homeomorphism-preserving mapping $M[P,t]$ from P to some subtree t of T such that $\lambda(|V_t| - |V_P|) + \sum_{\forall(u,v)\in M} \Delta[u,v]$ is maximal.

They solve the problem by bottom-up dynamic programming algorithm with runtime $O(\frac{|V_P|^2|V_T|}{\log|V_P|} + |V_P||V_T| \log|V_T|)$, where $|V_P|$ and $|V_T|$ are the number of vertices in pattern and text, respectively. They have developed the pathway alignment tool called as MetaPathway-Hunter.

3.3 MaWish

Koyuturk et al ([37]) introduce and employ a duplication /divergence model for evolution of PPI networks. Authors' solution allows to delete pattern and text vertices and different vertices in pattern and text can be mapped to one vertex respectively in text and pattern. They rebuild a alignment graph with $|V'| = O(|V_P||V_T|)$ vertices and $|E'| = O((|V_P||V_T|)^2)$ and proposed greedy heuristic error estimation algorithm. SAGA [58] converts graphs to fragment index first and calculated the difference of paths in pattern and text. Its greedy algorithm may obtain minimum subgraph distance. The heuristic greedy algorithms can not guarantee to find optimal solution.

3.4 Luonan Chen's approach

Li et al ([39]) formulate the problem as an integer quadratic problem to obtain the global similarity score based on the mapping of as many as node-to-node similarity and as many as edge-to-edge similarity. An exhaustive searching approach was employed in [61] to find the vertex-to-vertex and path-to-path mappings with the maximal mapping score under the condition of limited length of gaps or mismatches. The algorithm has the worst case time complexity $O(2^m \times m^2)$.

3.5 Path match and graph match

Q. Yang and S. Sze ([61]) focus on two problems : path matching and graph matching. Authors reduce the path matching problem to finding a longest weighted path in a directed acyclic graph and show that the problem of finding top k suboptimal paths can be solved in polynomial time. Their graph match approach reduces the problem to finding highest score subgraphs in a graph. They allow to delete disassociated vertices or induced subnetwork in query network and then align its remainder to target network by subgraph isomorphism. Their exact algorithm solves the problem when the query graph is of moderate size.

In their both approaches, authors omit that they allow different vertices in pattern can be associated with a vertex in text as well as a vertex in pattern can be associated multiple vertices in text graph. This permission is equivalent to allowing different vertices in query path can be mapped to a vertex in text in an optimal graph matching.

3.6 MetaPAT: S. Wernicke and F. Rasche's approach

S. Wernicke and F. Rasche ([56]) formulated the alignment as the maximum score embedding problem. Based on subgraph isomorphism, their embedding allows to stretch path or shorten path on both patten and text sides, which leads to homeomorphic embedding. Authors proposed a naive backtracking algorithm to exhaustively search for all simple paths and to obtain an optimal embedding. The algorithm takes exponential runtime even if the pattern is a tree. In order to reduce the search space, authors further propose to exploit local diversity which leads to the limited occurrences of consecutive gaps.

3.7 QNet and Torque : Sharon's approach

Sharan et al ([22]) formulate the problem as optimal homomorphism problem (Given a query $Q =< V_Q, E_Q >$ and a well-understood network $N =< V_N, E_N >$, find optimal homomorphism of $Q \rightarrow N$.). Authors employ color coding technique to mapping query tree to arbitrary graph by two steps. The first step is to randomly do coloration of pattern Q

which has 2^k possibilities. The second step is to compare the randomly generated colored graph with N. For any general query including multiple cycles, authors discussed their proposed solution which is based on tree decomposition of the query graph. The proposed algorithm for general query graph consists of three steps. The first one is to decompose the query graph to treelike graph, also called tree decomposition. The second step is to compare every tree-decomposition node to N. The third step is to merge the mapping results of different tree-decomposition nodes. Let T be the treewidth of the query graph, the runtime of their algorithm is $O(2^k \times |V_N|^{T+1})$.

Furthermore, Sharan et al ([12]) proposed an approach of network querying that does not rely on knowledge of the query topology. The problem is formulated to search a colored graph for connected subgraphs whose vertices have distinct given colors. Authors provide fixed-parameter algorithms that are based on the color-coding paradigm and dynamic programming (DP). In addition, authors provide an integer programming (ILP) formulation of the problem. The methods extended the work of QNet and can handle edge weights, insertions of network vertices (that do not match any query protein), and deletions of query nodes. The algorithm runs in $O(3^k \times |E_N|)$ time for handling the alignment without insertions and deletions of vertices.

However, the color-coding technique [4] can find optimal alignment with probability of $\frac{k!}{k^k} = e^{-k}$. Authors'a approaches are proposed for query graph. As a result, insertion of query nodes is not necessary, which are needed for the alignment to search for conserved pattern.

CHAPTER 4

OPTIMAL NETWORK ALIGNMENTS

Graph representation and methodology have gained popular applications in pattern recognition, software engineering, system security, machine learning, and scientific/medical informatics. The advantage of graphs results from the fact that they can be effectively used to represent structured data. Attributed graphs with an unrestricted label alphabet is one of the most general ways to define graphs. Metabolic pathways/networks, protein interaction network, and gene regulatory network can be represented as the graphs.

We first give graph theoretical models of pathways and then formulate the corresponding graph-theoretical problem.

4.1 Graph-theratical pathway models

First we describe the definition of graph and subgraph. Then we will present pathway models by graph representation.

4.1.1 Metabolic graph and metabolic subgraph

The following definitions are belonging to the general definition we will use in the dissertation proposal.

Definition 1. *A* ***graph*** *G is a four-tuple $G = (V, E, L_V, L_E)$ where*

1) V denotes a finite set of vertices.

2) $E \subseteq V \times V$ denotes a set of edges.

3) $L_V = \{l_V(v), v \in V\}$ denotes a set of vertex label alphabels.

4) $L_E = \{l_E(e), e \in E\}$ denotes a set of edge label alphabels.

l_V is a labeling function which is used to assign the vertex information into the label of the corresponding vertex in the graph. l_E is a labeling function which is used to assign the edge information into the label of the corresponding edge in the graph.

The graph definition includes several special cases. It gives the definition of not only the directed graph but also the undirected graph. For the undirected graph in which edges have no orientation, every edge $(u, v) \in E$ is not an ordered pair. However, for the directed graph, every edge $(u, v) \in E$ is an ordered pair. For the non edge attributed graphs, the edge label alphabets are defined by $L_E = \emptyset$.

Definition 2. *Let $G1 = (V1, E1, L1_V, L1_E)$ and $G2 = (V2, E2, L2_V, L2_E)$ be graphs. Graph $G1$ is a **subgraph** of $G2$, written $G1 \subseteq G2$, if*

- *$V1 \subseteq V2$.*
- *$E1 \subseteq E2$.*
- *$L1_V(u) = L2_V(u), \forall u \in V1$.*
- *$L1_E(u, v) = L2_E(u, v), \forall (u, v) \in E1$.*

Definition 3. *A **path** is a non-empty graph $P = (V, E)$ with the form of $V = x_0, x_1, ..., x_k$ and $E = x_0x_1, x_1x_2, ..., x_{k-1}x_k$, where x_i is all distinct.*

Definition 4. *If $P = x_0x_1...x_{k-1}$ is a path and $k \geq 3$, then the graph $C = P + x_{k-1}x_0$ is called a **cycle**.*

Definition 5. *Let $P = x_0x_1...x_k$ is a path and $k \geq 2$. If every intermediate vertex $x_1, ..., x_{k-1}$ in P has only degree 2, then the path P is called a **non-branching path**.*

Definition 6. *Let (u, v) be an edge of the graph $G = (V, E, L_V, L_E)$, the operation which adds a new vertex w and replace (u, v) with two new edges (u, w) and (w, v) is called **edge subdivision**.*

Informally speaking, the subdivision of the edge (u, v) is formed by putting a new vertex w anywhere in its interior. A subdivision of a graph $G1$ is a graph resulting from the subdivision of edges in $G1$.

4.1.2 Pathways

Biochemical pathway is a building block of metabolism which is a series of chemical reactions catalyzed by enzymes that occur within a cell. It represents a flow of consecutive enzymatic reactions that take specific metabolites (also called substrate) as an input to yield specific products. In such pathways, a substrate is converted into a product by the first enzyme in the pathway and the product of the first reaction then becomes the substrate for the next reaction. The sequence of reactions continue until the final product is made. When a biochemical pathway is functioning, the initial substrate is continually converted to the final product through the series of steps in the pathway. See Figure 4.1 as an example.

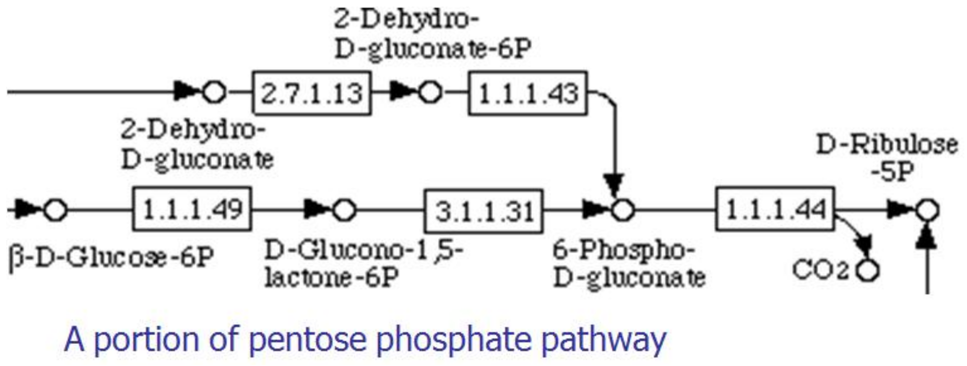


Figure 4.1. A portion of pentose phosphate pathway.

There exist two types of graph models representing metabolic pathways.

One model is based on hypergraphs. Metabolic networks can be represented as directed hypergraphs ([62]),where an edge, called a hyperedge, can connect more than two vertices. The vertices in the hypergraph are the compounds and the hyperedges are the reactions connecting the compounds. Since a reaction is treated as a single entity in a hypergraph, it can be used to capture relationships between any number of metabolites involved in a reaction [41]. The representation keeps the dependence between metabolites.

The other model is based on directed graphs in which vertices correspond to enzymes and there is a directed edge from one enzyme to another if the product of the reaction catalyzed by the first enzyme is a substrate of the reaction catalyzed by the second. See Figure 4.2 as an example. The directed graph reflect the flow of the metabolic chemical reactions, no matter whether the chemical reactions are reversible or not.

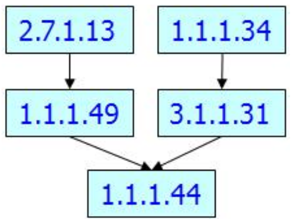


Figure 4.2. Graph model of a portion of pentose phosphate pathway.

In this dissertation proposal, we represent the metabolic network/pathways as directed graphs because enzyme is a type of protein and the representation easily expresses the genomics information.

4.2 Graph-theratical approaches to mappings

Existing mapping tools on this problem are mostly based on isomorphic and homeomorphic embeddings (see [53, 14, 40, 13, 25, 59, 50, 27, 5, 60, 21, 23, 35, 11] and [15, 16, 44, 43, 34, 33, 49, 39, 37, 55, 61, 19, 58]), effectively solving a problem that is NP-complete [26] even when searching a match for a tree in acyclic networks.

4.2.1 Exact graph matching

In exact graph matching, the goal is to determine if or not the labels and network topology, or part of them, of two graphs are identical.

If the network topology or graph structure of two graphs are linear (e.g. they can be represented as strings or feature vectors), the network alignment becomes similar to sequence alignment. Or else, when the vertices and edges can not be ordered in general, the problem becomes a class of graph isomorphism problem.

4.2.1.1 Graph isomorphism

Definition 7. *Let $G1 = (V1, E1, L1_V, L1_E)$ and $G2 = (V2, E2, L2_V, L2_E)$ be graphs. A* ***graph isomorphism*** *between $G1$ and $G2$ is a bijective function $f : V1 \rightarrow V2$ satisfying*
1) $l1_V(u) = l2_V(u), \forall u \in V1$.
2) $\forall (u, v) \in E1, \exists (f(u), f(v)) \in E2$ such that $L1_V(u) = L2_V(f(u))$ and $L1_V(v) = L2_V(f(v))$.
3) $\forall (u, v) \in E2, \exists (f^{-1}(u), f^{-1}(v)) \in E1$ such that $L1_V(f^{-1}(u)) = L2_V(u)$ and $L1_V(f^{-1}(v)) = L2_V(v)$.

Two graphs $G1$ and $G2$ are called isomorphic if there exists a graph isomorphism between them. The isomorphism of $G1$ and $G2$ means their labels and their network topologies are identical.

Closely related to graph isomorphism is the problem to detect if there exists a subgraph in the larger graph which is equal to the smaller graph. The problem is known as the one of finding subgraph isomorphism.

4.2.1.2 Subgraph isomorphism

Definition 8. *Let $G1 = (V1, E1, L1_V, L1_E)$ and $G2 = (V2, E2, L2_V, L2_E)$ be graphs. An injective function $f : V1 \rightarrow V2$ is called a* ***subgraph isomorphism*** *from $G1$ to $G2$ if there exists a subgraph $G \subseteq G2$ such that f is a graph isomorphism between $G1$ and G.*

A subgraph isomorphism exists from $G1$ to $G2$ if a subgraph of the larger graph is isomorphic to the smaller graph. The subgraph isomorphism is belonging to the class of NP-complete problem.

Another problem is known as maximum common subgraph problem.

4.2.1.3 Maximum common subgraph

Definition 9. *Let $G1 = (V1, E1, L1_V, L1_E)$ and $G2 = (V2, E2, L2_V, L2_E)$ be graphs. A graph $G = (V, E, L_V, L_E)$ is called a common subgraph of $G1$ and $G2$ if there exists subgraph isomorphism from G to $G1$ and from G to $G2$. If there exists no other common subgraph of $G1$ and $G2$ larger than G, the common subgraph G is called maximum common subgraph.*

Generally, the maximum common subgraph is not uniquely defined. There may exist multiple maximum common subgraph with a maximal number of vertices. The problem can be reduced to find the maximum clique problem in an association graph of two graphs. The association graph represents the set of vertex-to-vertex mappings that preserve the edge topology structures among all these corresponding vertices in both graphs.

4.2.2 Error-tolerant or inexact graph matching

In some real-world applications with graph patterns, it is often allowed that graphs from the same class differ according to the labels or/and the topology. The class of problems is classified as error-tolerant/inexact graph matching.

In error-tolerant or inexact graph matching, the goal is to determine if or not the labels and network topology, or part of them, of two graphs are similar. Graph correcting process which turns the different properties in both graphs to be the same should always lead to the exact graph matching.

When the noise in a class of graphs happens only in labels of graphs and not in the topology, error-correcting graph/subgraph isomorphism is required as a problem formulation to judge the similarity of graphs and to search similar graphs in a set of graphs.

If topologies are allowed to be different, homeomorphism is required as a problem formulation to judge the similarity of graphs and to search similar graphs in a set of graphs.

In the general case, graph edit distance has wider applications. In this subsection, we will describe homeomorphism and graph edit distance.

4.2.2.1 Homeomorphism

Definition 10. *Let $G1 = (V1, E1, L1_V, L1_E)$ and $G2 = (V2, E2, L2_V, L2_E)$ be graphs. Two graphs $G1$ and $G2$ are homeomorphism if some subdivision of $G1$ is isomorphic to some subdivision of $G2$.*

Two graphs $G1$ and $G2$ are homeomorphic if there is an isomorphism from some subdivision of $G1$ to some subdivision of $G2$. Edge subdivision is an operation that does not change homeomorphism type of a topological representation of a graph.

4.2.2.2 Graph edit distance

When the graph matching needs take topology error into account, graph edit distance offers an intuitive way to integrate error tolerance into the graph matching process. The key idea is to model topological changes by edit operations. The edit operations reflect the changes of topology and its processing. A set of standard edit operations consists of vertex insertion, vertex deletion, vertex substitution, edge insertion, edge deletion, and edge substitution operation. The substitution of vertex or edge is equivalent to the relabeling of the corresponding vertex or edge. The graph distance from a graph $G1$ to the other graph $G2$ is defined as the minimum cost taken over all sequences of edit operations that are necessary for the correction of the distortions in the graph $G2$. The smaller the edit distance between two graphs is, the more similar they are.

Definition 11. *Let $G1 = (V1, E1, L1_V, L1_E)$ and $G2 = (V2, E2, L2_V, L2_E)$ be graphs. The graph edit distance of $G1$ and $G2$ is defined by*

$$d(G1, G2) = min_{(\epsilon_1, \ldots, \epsilon_k) \in \wp(G1,G2)}(sum_{i=1}^{k} c(\epsilon_i))$$

where $\wp(G1, G2)$ denotes the set of edit paths transforming $G1$ into $G2$, ϵ denotes the corresponding edit operation, and c denotes the edit cost function

The edit distance is defined by the edit path with minimum accumulated edit operation costs. By using the appropriate parameters on the distortion and penalty cost model, it will be sufficient for similar graphs to turn one graph into the other. However for dissimilar graphs, even the minimum cost edit path will contain some strong distortion.

4.3 A novel formulation of the optimal network alignment

The widespread evolutionary machinery of gene duplication results in vertex copying [48]. The results of network alignments can be enhanced when gene duplication and divergence are taken into account. If two enzymes in the pattern species are evolutionarily related, the corresponding enzymes can be mapped into a single enzyme. The mapping explores the characteristics of graph homomorphism.

Additionally, there exist disassociated vertices or induced subnetworks which work as the noise and distract the graph matching. We allows to delete the disassociated part in query network and queried network.

Due to the above two reasons - the gene duplication and enzyme function sharing plus the existence of disassociated vertices or induced subnetwork, we can not directly formulate metabolic network alignments as one of the above problem formulations.

As a result, we integrate all characteristics of the above formulations (disregarding graph edit distance) and propose a new formulation of the optimal metabolic network alignment. First I will describe enzyme-to-enzyme similarity, vertex collapsing, path contraction, vertex deletion, and finally optimal network alignment.

4.3.1 Enzyme-to-enzyme similarity

We provide two approaches to calculate vertex similarity score. One approach is to employ the lowest common upper class distribution proposed by Tohsato et al and discussed in [43]. The corresponding penalty score for gap is 2.0.

The other approach is proposed by Dr. Robert Harison. The approach is to make full use of EC encoding and the tight reaction property classified by EC. The EC number is expressed with a 4-level hierarchical scheme. The 4-digit EC number, $d_1.d_2.d_3.d_4$ represents a sub-sub-subclass indication of biochemical reactions. The corresponding penalty score for gap is 0.5.

Given two enzyme $u = d_1.d_2.d_3.d_4$ and $v = d'_1.d'_2.d'_3.d'_4$, a **similarity cost score** Δ is a nonnegative real number. It is defined by:

1. $\Delta(u, v) = \infty$, if $d_1 \neq d_1$';
2. $\Delta(u, v) = \infty$, if $d_1 = d_1$' and $d_2 \neq d_2$';
3. $\Delta(u, v) = 10$, if $d_1 = d_1$' and $d_2 = d_2$' and $d_3 \neq d_3$';
4. $\Delta(u, v) = 1$, if $d_1 = d_1$' and $d_2 = d_2$' and $d_3 = d_3$' and $d_4 \neq d_4$';
5. $\Delta(u, v) = 0$, if $d_1 = d_1$' and $d_2 = d_2$' and $d_3 = d_3$' and $d_4 = d_4$';

4.3.2 Vertex collapsing

Due to gene duplication and function sharing, we propose to additionally relax one-to-one correspondence between vertices – instead we allow different pattern vertices to be mapped to a single text vertex. The relaxation is called vertex collapsing. Such relaxation may sometimes cause confusion – a path can be mapped to a cycle. For instance, if two enzymes with similar functions belong to the same path in the pattern and a cycle with similar enzymes belongs to the text, then the path can be mapped into a cycle (see Figure 4.3). However, if the text graph is acyclic this cannot happen. Even if there are cycles in the text, still one can expect that functionally similar enzymes are very rare in the same path.

4.3.3 Path contraction

The obvious way to preserve the pathway topology is to use isomorphic embedding – one-to-one correspondence between vertices and edges of the pattern and their images in the

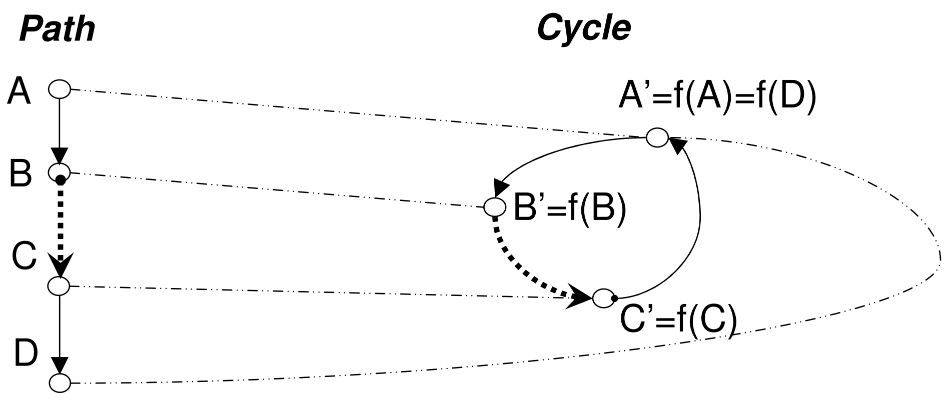


Figure 4.3. Alignment of a path $(A, B, \ldots, C, D)$ in the pattern with a cycle $(A', B', \ldots, C', D' = A')$ in the text.

text. The requirement on edges can be relaxed – an edge in the pattern can be mapped to a path in the text ([44, 43]). The relaxation is called path contraction. [44, 43] only allow the occurrences in the text.

Definition 12. *Let $P = x_0 x_1 ... x_k$ is a non-branching path in the graph $G = (V, E, L_V, L_E)$, path contraction is to replace the path P with an edge (x_0, x_k).*

Furthermore the path contraction can happen in both the pattern and the text. The corresponding mapping is called a *homeomorphism*.

4.3.4 Vertex deletion

For keeping the order of vertices such that for any vertex v all children cannot communicate with each other except through v, we assume that all deleted vertices follow the same order. If the order of vertices u and v is $u < v$, the deletion order can not be $v < u$. So it's child branch which has more than 1 vertics and is an induced subgraph has been visited before it.

As a consequence, we allow pattern vertex deletion in the following constraints (also see fig. 4.4 as an example):

(1) for degree only one vertex, delete the vertex and adjacent edge;

(2) for degree two vertex, delete the vertex and adjacent two edges and then add an extra edge connecting the endpoints of two edges;

(3) Following the order of visiting pattern vertex, (1) and (2) could repetitively happen so that a submodule may be allowed to delete (called as a strong deletion);

(4) for vertex b with degree larger than 2, only when its child branches have been deleted so that a degree-2 vertex is left, (2) constraint can be considered or else the vertex will not be allowed to delete (called as a bypass deletion).

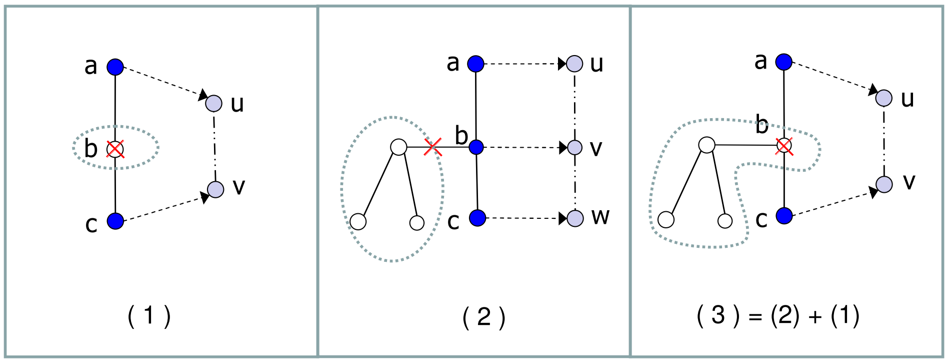


Figure 4.4. Examples of pattern vertex deletion. Solid lines represent pattern edges; dashed lines represent text paths; dashed arrows connect pattern vertices with their images in the text graph. (1) Bypass deletion of a patten vertex of degree 2. (2) Branch strong deletion of three pattern vertices; (3) =(2)+(1) Composition of strong and bypass deletions: after strong deletions a pattern vertex becomes eligible to bypass deletion.

4.3.5 An optimal network alignment

4.3.5.1 Basics

Given two sets A and B, a **binary relation** R from A to B is a subset of Cartesian product $A \times B = \{(a, b)|a \in A, b \in B\}$. If $R \subseteq A \times B$ and $(a, b) \in R$, we say that a is related to b by R. The **domain** of R, $Dom(R)$, consists of elements in A that are related to elements in B. Namely, $Dom(R) = \{a \in A|\exists b, (a, b) \in R\}$. The **range** of R, $Ran(R)$, consists of elements in B that are second elements of pairs in R. Also we have $Ran(R) = \{b \in B|\exists a, (a, b) \in R\} \subseteq B$.

A binary relation R from A to B is a **function** denoted as $f(a) = b$ if $|\{b \in B|(a,b) \in R\}| = 1$.

A **partial function** f from A to B, $f : A \rightarrow B$, is a function from a subset of A to B, satisfying $\forall a \in A$, $|\{b \in B|(a,b) \in R\}| \leq 1$.

A partial function f from A to B is a **1-to-1 partial function** if $f(a^{'}) \neq f(a)$ for two distinct elements a and $a^{'}$ and $|\{f(a^{'})\}| = 1$ & $|\{f(a)\}| = 1$.

4.3.5.2 Definitions and notations

Let graph $G =< V_G, E_G >$, and graph $H =< V_H, E_H >$, let us give the following definitions.

A function $f : V_G \rightarrow V_H$ is homomorphism h from G to H if $\forall (u,v) \in E_G, u, v \in Dom(f)$, $(f(u), f(v)) \in E_H$.

A homomorphism h from G to H is called 1-to-1 homomorphism if h is an 1-to-1 function.

A partial homomorphism $h^{'}$ from G to H is a homomorphism from a subset of G to H.

The partial homomorphism is one to one such that if $h_{v'}(v) = h_{v'}(u)$, then $v = u$.

1-to-1 homomorphism is called embedding. 1-to-1 partial homomorphism is called partial embedding.

For any graph G, its transitive closure is $G^* =< V, E_{G*} >$, in which any $e = (u,v) \in E_{G*}$ is a path from u to v in G.

An embedding from G to the transitive closure of H is called a homeomorphic mapping $hf : G \rightarrow H$.

A bypass branch in graph from u to v is a specific u-v path such that any vertex except u and v has degree 2. The intermediate vertices between u and v are called bypass vertices. The set is denoted as B.

Bypass deletion $d : G \rightarrow H$ is a partial embedding from G to H which satisfies $V_G - Dom(d) \in B$ and $\forall u, v \in Dom(d)$, if there exist a path from u to v, there is an edge $(d(u), d(v))$ in H.

Partial homeomorphic embedding consists of the composition $d \circ hf$ of bypass deletion d and homeomorphism mapping hf.

Our specifical **homo-home morphism** $HH : G \rightarrow H$ is the composition of several relations as follows

(1) $h^{'} : G \rightarrow G'$, partial homomorphism from G to G';

(2) $d : G' \rightarrow G"$, bypass deletion from $G' \rightarrow G"$;

(3) $hf : G" \rightarrow H$, homeomorphism mapping from $G" \rightarrow H$;

So

$$HH : G \xrightarrow{h^{'}} G' \xrightarrow{d} G" \xrightarrow{hf} H$$

4.3.5.3 Problem Formulation

The **cost of homo-home morphism** $HH : G \rightarrow H$ is

$$cost(f) = \sum_{v \in V_G \ \& \ HH(v) \neq \emptyset} \Delta(v, f(v)) + \sum_{v \in V_{G'} \ \& \ d(v) = \emptyset} (\Omega(v)) + \lambda \sum_{e \in E_{G"} \ \& \ |hf(e)| \geqslant 1} (|hf(e)| - 1)$$

Given a graph P and a graph H, to find minimum cost of homo-homeo morphism $HH : G \rightarrow H$.

CHAPTER 5

NETWORK ALIGNMENTS FROM A POLYTREE TO AN ARBITRARY GRAPH

A metabolic pathway is a series of chemical reactions catalyzed by enzymes that occur within a cell. Metabolic pathways are represented by directed networks in which vertices correspond to enzymes and there is a directed edge from one enzyme to another if the product of the reaction catalyzed by the first enzyme is a substrate of the reaction catalyzed by the second.

Mapping metabolic pathways should capture the similarities of enzymes represented by proteins as well as topological properties that cannot be always reduced to sequential reactions represented by paths. Below we first describe our approach to measure enzyme similarity.

Our implementation provides two alternative enzyme similarity scores. One approach is to employ the lowest common upper class distribution proposed in [52] and discussed in [43], which similarity score of two enzymes $E1$ and $E2$ is equal to log_2^{-h} where h is the number of enzymes belonging to the lowest common enzyme class of $E1$ and $E2$. The corresponding penalty score for gap is 2.0.

The other approach is a novel enzyme-to-enzyme dissimilarity score scheme Δ based on 4-level EC encoding d1:d2:d3:d4, where d4 is the number of enzymes in d3-class which is subclass of d2-class which is subclass of d1-class. The numeric values are usually positive integers and equal 0 (or -) only if the corresponding subclass is unknown. If d1's or d2's of two enzymes are different and non-zero, then $\Delta = \infty$, otherwise, if d3's are different and non-zero, then $\Delta = 10$, otherwise if d4's are different and non-zero, then $\Delta = 1$, and, otherwise, $\Delta = 0$ (see [19]). Notice that if two enzymes are different in the corresponding position but one of them is 0, then we do not increase Δ. This is because we do not penalize the lack of information.

The topology of most metabolic pathways is a simple path, but frequently pathways may branch or have several incoming arcs – all such topologies are instances of a *polytree*, i.e., a directed graph which becomes an undirected tree when edge directions are disregarded. The query pathways are usually simple and can be represented as a polytree but in some cases they can have a cycle or alternative ways to reach the same vertex. We will suggest to follow the standard practice of breaking such cycle or paths by removing edges in chapter 6.

5.1 Definitions and notations

We distinguish two kinds of vertex deletions: bypass deletion and strong deletion. A *strong deletion* of a vertex v is removing of arbitrary vertex v together with all its incoming and outgoing edges which is formally represented as mapping into $\mathbf{d}$. A *bypass deletion* of a vertex v with a single incoming and a single outgoing edge is replacing of a u-v-w-path with a single (u, w)-edge formally represented as mapping v into $\mathbf{b}$.

Let graphs $P = (V_P, E_P)$ and $T = (V_T, E_T)$ represent a pattern and text metabolic networks, respectively. Let $f : P \to T$ map every vertex in V_P to $V_T \cup \{\mathbf{b}, \mathbf{d}\}$. A pair of pattern vertices $u, v \in V_P$ is called *contracted* if u and v are mapped into two different text vertices and either $(u, v) \in E_P$ or there exists a u-v-path in P whose all intermediate vertices are bypass deleted. Let E_P^f be the set of all vertex pairs in P contracted by f (note that $E_P \subseteq E_P^f$). The mapping f is called a *network alignment* if for each contracted pair $(u, v) \in E_P^f$ there exists a $f(u)$-$f(v)$-path in the text T. The number insertions, i.e., the minimum number of intermediate vertices in such a path is denoted $\sigma(f(u), f(v))$.

The alignment should be penalized (i) for mismatches between aligned enzymes, (ii) for strong deletions, (iii) for bypass deletions, and (iv) for insertions. Thus we obtain the following cost function.

$$cost(f) = \sum_{u \in V_P} \Delta(u, f(u)) + \lambda \sum_{(u,v) \in E_P^f} \sigma(f(u), f(v)),$$

where $\Delta(u, f(u))$ is the penalty of mismatch between enzymes corresponding to pattern vertex u and text vertex $f(u)$, $\Delta(u, \mathbf{d})$ and $\Delta(u, \mathbf{b})$ are penalties for strong and bypass deletions, respectively, and λ is the penalty for a single enzyme insertion.

Given two metabolic networks P and T, the problem is to find the optimal (minimum-cost) network alignment from P to T.

5.2 Network alignments from a polytree to an arbitrary graph

A polytree is a directed graph, whose underlying undirected graph is a tree. Aligning between a polytree as the pattern and an arbitrary graph as the text is an extension of the tree edit distance problem (see [17]). The solution is a combinatorial algorithm based on the classic technique of dynamic programming. Under the consideration of gene duplication and function sharing, our algorithm holds the assumption that a childs contribution to their parents mapping are independent to another childs contribution. As a result, for any vertex u_i as a child of u in the pattern, there exists three possibilities to make the contribution to its parent's alignment, e.g.$u \rightarrow v$ (v in the text):

- u_i is mapping to v_j (v_j is a descendent of v)
- u_i is strongly deleted (denoted by strongD(u_i))
- internal vertices denoted as bypass(u,u_i,u_t) running between u and u_t and across u_i are bypassed after their descendants are strongly deleted (denoted by weakD(u, u_i, u_t))

Given a cost function $\Delta(u, v)$ defined on a pair of vertices $u \in V_P$ and $v \in V_T$, we build two dynamic tables A and B, in both of which each row and column of this table corresponds to a vertex of P and T, respectively. While the columns are in no particular order, the rows are sorted according to the opposite of the DFS traversal of P. As a result, for any vertex u all children cannot communicate with each other except through u (Namely, each edge e_i in

P is the unique edge connecting u_i with the previous vertices in the order). Each element $A[u, v]$ corresponds to the best cost of a alignment from the subgraph of P induced by the vertices previous to u into T which maps u into v. Each element $B[u_i, v]$ corresponds to the best contribution of u_i to a alignment from the u to v. Every table is filled from the bottom to the top. Initially,

$$A(u,v) = \begin{cases} \min\{\delta(u,v), strongD(u)\} & \text{when vertex u is leaf} \\ \infty & \text{otherwise} \end{cases}$$

and $B(u, v) = \infty$. Then both tables are filled iteratively in a butterfly way by

$$B[u_i, v] = \min \begin{cases} \min_{v_j \in child(v)}(A(u_i, v_j) + \lambda h(v, v_j)) \\ strongD(u_i) \\ \min_{u_{i,k} \in des(u_i)}(weakD(u, u_i, u_{i,k}) + \lambda h(v, v_j) + A(u_{i,k}, v_j)) \end{cases}$$

and

$$A(u,v) = \Delta(u,v) + \sum_{u_i \in child(u)} B(u_i, v)$$

where $h(v, v_j)$ is the number of hops from v to v_j, λ is the penalty per an insertion, child(v) is the set of all direct successor of v, and des(u_i) is the set of all successor of u_i. Penalty for strong deletion and weak deletion can be computed in a preprocess by

$$strongD(u) = \Delta(u, d) + \sum_{u_{i*} \in des(u)} \Delta(u_{i*}, d)$$

and

$$weakD(u, u_i, u_t) = \sum_{u_k \in bypass(u, u_i, u_t)} \left(\sum_{u_{k,j}} strongD(u_{k,j}) + \Delta(u_k, b)\right)$$

In the preprocess, we have used two queue structures to compute weakD table under the DFS traversal of the pattern and obtain the transitive closure of the text.

Our dynamic programming algorithm (see [17] for details) aligns polytrees with arbitrary networks in time $O(|V_P|^2 |E_T| \log |V_T|)$.

5.3 Preprocessing

In the preprocessing, we first describe text graph preprocessing; then we present pattern graph ordering; finally we describe how to calculate the penalties of pattern vertex strong deletions and of pattern vertex weak deletions.

5.3.1 Transitive closure of the text graph

In order to compute the cost of vertex insertion, it is necessary to know the number of hops for any shortest path in the text graph T. Essentially, the transitive closure of a directed graph contains an edge for every path in the graph.

Definition 13. *The* ***transitive closure*** *of a directed graph $G = (V, E)$ is a directed graph $T_c(G) = (V, E_c)$ where $E \subseteq E_c$ and for every pair of vertices $u, v \in V$, if there is a path from u to v in G, then there is an edge $(u, v) \in E_c$.*

Although finding single-source shortest paths in general graphs is slow, in our case it is sufficient to run breadth-first-search with runtime $O(|E_T| + |V_T|)$. Assuming that T is connected, i.e., $|E_T| \geq |V_T|$, we conclude that the total runtime of finding all shortest paths is $O(|V_T||E_T|)$. In the resulting transitive closure $T' = (V_T, E'_T)$ of the graph T, each edge $e \in E'_T$ is supplied with the number of hops $h(e)$ in the shortest path connecting its ends.

5.3.2 Pattern Graph Ordering

We will further need a certain fixed order of vertices in V_P as follows. Let $P' = (V_P, E'_P)$ be the undirected tree obtained from P by disregarding edge directions. Let us choose an arbitrary vertex $r \in V_P$ as a root and run depth-first search (DFS) in P' from r. Let $\{r = v_1, \dots, v_{|V_P|}\}$ be the order of the DFS traversal of V_P and let $e'_i = (v_i, v) \in E'_P$ (corresponding to directed edge $e_i \in E_P$) be the unique edge connecting v_i to the set $\{v_1, \dots, v_{i-1}\}$. The vertex $v \in \{v_1, \dots, v_{i-1}\}$ is called a *parent* of v_i and v_i is called a *child* of v.

5.3.3 Calculation of the penalties for pattern vertex deletions

First we will describe the definition of pattern vertex deletions, and then we will present our approach to calculate the penalties of the deletions.

5.3.4 Pattern vertex deletions

We distinguish two kinds of vertex deletions: (i) bypass deletion corresponding to the replacement of a few consecutively acting enzymes with a single multifunctional enzyme or enzyme using an alternative catalysis and (ii) strong deletion symbolizing the matching of a proper connected subgraph of the pattern network. Thus, a *bypass deletion* of a vertex v with a single incoming and a single outgoing edge is the replacing of a u-v-w-path with a single (u, w)-edge formally represented as mapping v into **b**. A *strong deletion* of a vertex v is removing of arbitrary vertex v together with all its incoming and outgoing edges which is formally represented as mapping into **d** (see 4.4).

The specified *bypass deletion* of a vertex of degree 2 supports homeomorphism, i.e., it allows the replacement of a path with a single edge. If the pattern is a directed graph, then a vertex v can be bypassed only if it belongs to a directed path $a \rightarrow v \rightarrow b$, and consequently the incoming and outgoing edges are replaced by a single edge $a \rightarrow b$. Both types of deletion can be applied recursively and together with each other.

The specified *strong deletion* corresponds to the operation of deleting subgraph of the pattern and obtaining a vertex-induced subgraph. A vertex-induced subgraph P', sometimes simply called an "induced subgraph", is a subset of the vertices of a graph P together with any edges whose endpoints are both in this subset. Namely, if a reachable path in the graph G run across any two vertices in P', the path should be reserved in P'.

The combination of the two types of pattern vertex deletions is defined as weak deletion.

We have defined two queues and a couple of recursive processes to calculate the pattern vertex deletion penalties.

5.4 Dynamic programming approach

In this section we first describe a novel fast algorithm for finding optimal network alignment of the tree patterns with no deletions, then of the tree patterns with deletions, and finally, of arbitrary patterns with limited vertex feedback set.

5.4.1 Tree patterns with no deletions

We orient the undirected graph of the pattern so it is a rooted tree with a root r and each node u has a set of children. If the pattern is undirected, $\bar{\sigma}$ is the same as σ, otherwise if v is a child of u in our rooted tree

$$\bar{\sigma}(f(u), f(v)) = \begin{cases} \sigma(f(u), f(v)) & \text{if} \quad (u, v) \in E_P \\ \sigma(f(v), f(u)) & \text{if} \quad (v, u) \in E_P \end{cases}$$

We can define cost of f restricted to the subtree rooted at u, so $\mathit{cost}(f, r) = \mathit{cost}(r)$. If u is a leaf, $\mathit{cost}(f, u) = \Delta(u, f(u))$. Otherwise

$$\mathit{cost}(f, u) = \Delta(u, f(u)) + \lambda \sum_{\text{child } v \text{ of } u} \mathit{cost}(f, v) + \bar{\sigma}(f(u), f(v))$$

Now we define the following two recursive functions $A, B : V_P \times V_T \to \mathbb{R}$. In the algorithm, we fill two dynamic programming tables with their values.

$A(u, x)$ is defined as the least value of $\mathit{cost}(f, u)$ such that $f(u) = x$. Note that the optimum solution cost is $\min_{v \in V_T} A(r, v)$.

$B(v, x)$ is defined as the least value of $A(v, y) + \bar{\sigma}(x, y)$, i.e., the contribution that child v can give to the cost of its parent u if $f(u) = x$. Having an additional table for B accelerates the computation of A, because it is faster to compute $B(v, x)$ for each x together than separately.

If u is a leaf or $B(v, *)$ is computed and tabulated for every child v of u, we apply the formula

$$A(u, x) = \Delta(u, x) + \sum_{\text{child } v \text{ of } u} B(v, x)$$

Computing $A(u, x)$ takes time proportional to the number of children of u therefore the total time spend computing the values of A is

$$O(|V_T|(\sum_{u \in V_P} deg(u))) = O(|V_T||E_P|) = O(|V_P||V_T|)$$

Computation of $B(v, x)$ is more involved. Implementation proposed in [19] requires computation of the transitive closure $T' = (V_T, E'_T)$ of the text graph T. Computing

$$B(v, x) = \min_{(x,y) \in E'_T} (A(v, y) + \bar{\sigma}(x, y))$$

takes $O(|V_P||E'_T|)$ runtime which can be as large as $O(|V_P||V_T|^2)$.

We propose instead the following adaptation of Dijkstra algorithm. Given the values $A(v, y)$ for every $y \in V_T$, it finds the values of $B(v, x)$ for all $x \in V_T$ using a priority queue Q. The pseudo-code below assumes, without loss of generality, that the edge connecting v with its parent u is directed (u, v). An item (w, k) in Q is a node w with priority key k.

```
for each $x \in V_T$
    insert $(x, A(v, x) - \lambda)$ into $Q$
    $B(v, x) \leftarrow \infty$
while $Q$ is not empty
    delete from $Q$ item $(y, k)$ with the minimum key $k$
    for every $(x, y) \in E_T$
        if $B(v, x) > k + \lambda$
            $B(v, x) \leftarrow k + \lambda$
        if $y > k + \lambda$
```

```
            decrease key of $x$ in $Q$ to $k + \lambda$
```

If we implement the priority queue Q with the Fibonacci heaps, the runtime for computing $B(v, x)$ for all $x \in V_T$ is $O(E_T + V_T \log V_T)$. Finally, the optimal $cost(f) = \min_v A(r, v)$ can be computed in time $O(|V_P|(|E_T| + |V_T| \log |V_T|))$.

A more practical priority queue based on a binary heap results in slightly higher total runtime of $O(|V_P||E_T| \log |V_T|)$.

5.4.2 Tree patterns with deletions

Handling the case with deletions does not increase the asymptotic running time, but it requires several additional considerations. To reduce the number of cases, we will assume that bypass deletion is applied only to so-called path nodes, pattern nodes that have degree 2 in the case of undirected pattern, or in- and outdegree 1 in the case of directed pattern (this considerably simplifies the enforcement of the last consistency rule).

For $u \in V_P$, let $D(u)$ be the sum of $\Delta(v, \mathbf{d})$ over all descendants v of u (the cost of strongly deleting the subtree of u). Note that the optimum f has some $u \in f^{-1}(V_T)$ such that $f^{-1}(V_T)$ is contained in the subtree of u; under that assumption the optimum cost equals $A(u, f(u)) + D(r) - D(u)$, so get the optimum cost by finding the minimum value of $A(u, x) + D(r) - D(u)$.

Moreover, when we consider the minimum contribution of a child, the value $B(v, x)$, we have to consider two new possibilities. One is that the entire subtree of the child is strongly deleted, so $D(v)$ is a possible value; this can be handled by initializing $B(v, x)$ with $D(v)$ rather than ∞. The second is that the child v bypass deleted, which means that it is a path node, and in the tree of P it has a single child w. In that case the contribution is "created" in the subtree of w and its cost is increased by $\Delta(v, \mathbf{b})$. To handle that, we introduce another function/array $C(v, x)$, which for non-path nodes equal $A(v, x)$, and for a path node v with a single child w equals $\min(A(v, x), C(v, x) + \Delta(v, \mathbf{b}))$, and we use $C(v, x) - \lambda)$ as the initial priority key of x (rather than $A(v, x) - \lambda)$.

5.5 Computational complexity analysis

As we mentioned earlier, the runtime for constructing the transitive closure $T' = (V_T, E'_T)$ is $O(|V_T||E_T|)$. the preprocess helps handle pattern vertex deletion, which does not increase the runtime in asymptotic mode.

The runtime to fill a cell $DT[i, j]$ is proportional to

$$t_{ij} = deg_P(v_i) deg_{T'}(u_j)$$

where $deg_P(v_i)$ and $deg_{T'}(u_j)$ are degrees of v_i and u_j in graphs P and T', respectively. Indeed, the number of children of v_i is $deg_P(v_i) - 1$ and for each child v_{i_l} of v_i there are at most $deg_{T'}(u_j)$ feasible positions in T' since $f(v_i)$ and $f(v_{i_l})$ should be adjacent. The runtime to fill the entire table DT is proportional to

$$\sum_{j=1}^{|V_T|} \sum_{i=1}^{|V_P|} t_{ij} = \sum_{j=1}^{|V_T|} deg_{T'}(u_j) \sum_{i=1}^{|V_P|} deg_P(v_i) = 2|E'_T||E_P|$$

Thus the total runtime is $O(|V_T||E_T| + |E'_T||V_P|)$. Even though T is sparse, $|E'_T|$ may be as large as $O(|V_T|^2)$, i.e., the runtime is $O(|V_T|(|E_T| + |V_T||V_P|))$.

5.6 Experimental data

The genome-scale metabolic network data in our studies were drawn from BioCyc [2, 32, 38], the collection of 260 Pathway/Genome Databases, each of which describes metabolic pathways and enzymes of a single organism. We have chosen metabolic networks of *E. coli*, the yeast *S. cerevisiae*, the eubacterium *B. subtilis* and the archeabacterium *T. thermophilus* so that they cover major lineages Archaea, Eukaryotes, and Eubacteria. The bacterium *E. coli* with 113 pathways is the most extensively studied prokaryotic organism. *T. thermophilus* with 208 pathways belongs to Archaea. *B. subtilis* with 226 pathways is one of the best understood Eubacteria in terms of molecular biology and cell biology. *S. cerevisiae* with 151 pathways is the most thoroughly researched eukaryotic microorganism.

5.7 Computation of statistical significance

Although the cost of an alignment reflects the similarity of pathways, it alone cannot assure us that such cost is not obtained by chance. Only statistically significant cost values can be taken in account. Statistical significance is measured by p-value, i.e., the probability of the null hypothesis that the cost value is obtained by pure chance. Following a standard randomization procedure, we randomly permute pairs of edges (u, v) and (u', v') if no other edges exist between these 4 vertices u, u', v, v' in the text graph by reconnecting them as (u, v') and (u', v). This allows us to keep the incoming and outgoing degree of each vertex intact. We find the minimum cost network alignment of the pattern graph with the full randomization of the text graph and check if its cost is at least as big as the minimum cost before the randomization of the text graph. We say that the alignment is statistically significant with $p < 0.01$ if we found at most 9 better costs in 1000 randomization of the text graph.

5.8 Experiment results

For each pair of four species (*B. subtilis*, *E. coli*, *T. thermophilus* and *S. cerevisiae*), using our algorithm we find the best alignment of each pathway of one species with each pathway of the other and check if this alignment is statistically significant, i.e., if $p < 0.01$. We have run our experiments on a Pentium 4 processor, 2.99 GHz clock with 1.00 GB RAM. The total runtime was 1.5h for the input/output of pathways and computing the optimal pattern-to-text mapping and its p-value for every pair of pathways (there are in total 516052 pattern-text pathway pairs).

5.8.1 Number of significant alignments.

The results of our experiments are reported in Table 5.1. The first column contains the name of the species from whose metabolic network the pattern pathways have been chosen. Note that if a pathway is not a polytree or degenerate (i.e., has less than 3 nodes), then it is

Table 5.1. Number of significant pair-wise alignments between metabolic pathways of *T. thermophilus*, *B. subtilis*, *E. coli* and *S. cerevisiae.*

pattern network (tree pathways)	# of mappings	text network (number of pathways)			
		T. thermophilus	*B. subtilis*	*E. coli*	*S. cerevisiae*
T. thermophilus (208)		38	21	18	18
	for pattern pathways	28	14	12	13
	for text pathways	35	20	18	17
B. subtilis (226)		162	217	121	58
	for pattern pathways	80	143	85	39
	for text pathways	106	153	92	40
E. coli (113)		9	5	38	3
	for pattern pathways	2	3	3	2
	for text pathways	9	5	14	5
S. cerevisiae (151)		24	12	12	14
	for pattern pathways	9	7	6	13
	for text pathways	21	12	12	14

Table 5.2. The list of all 20 pathways in *B. subtilis* that have statistically significant alignment images simultaneously in all 3 other species *E. coli*, *T. thermophilus* and *S. cerevisiae.* The lower part contains 4 more different pathways with statistically significant images in all 4 species.

Pathway name
alanine biosynthesis I
biotin biosynthesis I
coenzyme A biosynthesis
fatty acid beta
fatty acid elongation saturated
formaldehyde oxidation V (tetrahydrofolate pathway)
glyceraldehyde 3 phosphate degradation
histidine biosynthesis I
homoserine biosynthesis
lysine biosynthesis I
ornithine biosynthesis
phenylalanine biosynthesis I
phenylalanine biosynthesis II
polyisoprenoid biosynthesis
proline biosynthesis I
quinate degradation
serine biosynthesis
superpathway of gluconate degradation
tyrosine biosynthesis I
UDP galactose biosynthesis
alanine biosynthesis
biotin biosynthesis
fatty acid oxidation pathway
fructoselysine and psicoselysine degradation

Table 5.3. The list of 14 pathways conserved across *B. subtilis*, *E. coli*, and *T. thermophilus*; 2 more pathways conserved across *B. subtilis*, *E. coli*, and *S. cerevisiae*; 2 more pathways conserved across *B. subtilis*, *T. thermophilus*, and *S. cerevisiae.*

Pathway name
triple: *B. subtilis*, *E. coli*, and *T. thermophilus*
4 aminobutyrate degradation I
de novo biosynthesis of pyrimidine deoxyribonucleotides
de novo biosynthesis of pyrimidine ribonucleotides
enterobacterial common antigen biosynthesis
phospholipid biosynthesis I
PRPP biosynthesis II
salvage pathways of pyrimidine deoxyribonucleotides
ubiquinone biosynthesis
flavin biosynthesis
glycogen biosynthesis I (from ADP D Glucose)
L idonate degradation
lipoate biosynthesis and incorporation I
menaquinone biosynthesis
NAD biosynthesis I (from aspartate)
triple: *B. subtilis*, *E. coli*, and *S. cerevisiae*
oxidative branch of the pentose phosphate pathway
S adenosylmethionine biosynthesis
triple: *B. subtilis*, *T. thermophilus*, and *S. cerevisiae*
tyrosine biosynthesis I
fatty acid elongation unsaturated I

omitted. We did not omit any pathway from the text species since our algorithm supports any network as a text. For every species-to-species mapping, we compute the number of mapped pairs with $p < 0.01$, the number of the pattern pathways that have at least one statistically significant homomorphic image and the number of the text pathways that have at least one statistically significant homomorphic pre-image.

For example, for homomorphism from *T. thermophilus* to *B. subtilis*, there are in total 21 statistically significant mapped pairs, 14 non-degenerate tree *T. thermophilus* pathways have statistically significant homomorphic images in *B. subtilis* and 20 out of 226 *B. subtilis* pathways have statistically significant homomorphic pre-images.

5.8.2 Identifying Conserved Pathways.

We have identified the pathways that are conserved across all 4 species under consideration. Table 5.2 contains a list of all 20 pathways in *B. subtilis* that have statistically significant optimal alignment images simultaneously in all species. The lower part of Table 5.2 contains 4 more pathways with different names in *E. coli*, *T. thermophilus* and *S. cerevisiae*, which have simultaneous statistically significant images in all species.

Besides 24 pathways conserved across all 4 species we have also found 18 pathways only common for triples of these species. Table 5.3 gives the pathway names for each possible triple of species (the triple *E. coli*, *T. thermophilus* and *S. cerevisiae* does not have extra conserved pathways).

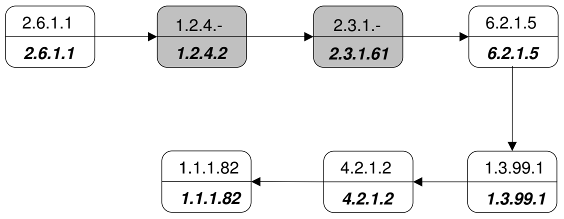


Figure 5.1. Mapping between glutamate degradation VII pathways of from *B. subtilis* to *T. thermophilus* ($p < 0.01$). The node with upper part and lower part represents a vertex-to-vertex mapping. The upper part represents the query enzyme and the lower part represents the text enzyme. The shaded node reflects enzyme homology.

5.8.3 Resolving Ambiguity.

Currently multiple pathways contain unresolved enzymes. Completely unresolved enzymes have EC notation 0.0.0.0/-.-.-.- and partially unresolved enzymes have less ”-”’s, e.g., EC notation 1.2.4.-. We can use our mapping tool to suggest possible resolution of these ambiguities as follows.

Let us consider two examples of network alignment – the mapping of glutamate degradation VII pathway in *B. subtilis* to glutamate degradation VII pathway in *T. thermophilus* (shown in Figure 5.1), and the mapping of interconversion of arginine, ornithine and proline

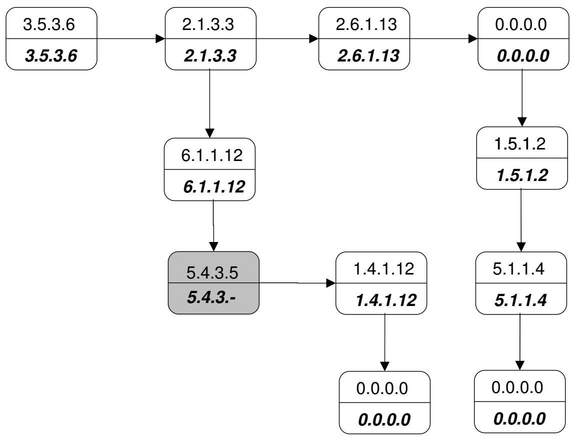


Figure 5.2. Mapping of interconversion of arginine, ornithine and proline pathway from *T. thermophilus* to *B. subtilis* ($p < 0.01$). The node with upper part and lower part represents a vertex-to-vertex mapping. The upper part represents the query enzyme and the lower part represents the text enzyme. The shaded node reflects enzyme homology.

pathway in *T. thermophilus* to interconversion of arginine, ornithine and proline pathway in *B. subtilis* (shown in Figure 5.2). When some enzymes in pathway are labeled with the end of ".-", it denotes their exact reactions are not explicit. The mapping results indicate that a potential similar enzyme with similar functions of the unclear enzyme can be found in some other species.

5.8.4 Identifying pathway holes

Pathway holes happen when a genome appears to lack the enzymes needed to catalyze reactions in a pathway [28]. We can use our mapping tool to identify potential pathway holes as shown in the following example (see Figure 5.3).

There is a statistically significant mapping from formaldehyde oxidation V (tetrahydrofolate pathway) in *B. subtilis* to formyITHF biosynthesis in *E. coli.* The correspondence between enzymes shows a gap – enzyme 3.5.1.10 is missing. We found that the enzyme 3.5.1.10 exists in *B. subtilis* according to Enzyme and Swiss-prot databases.

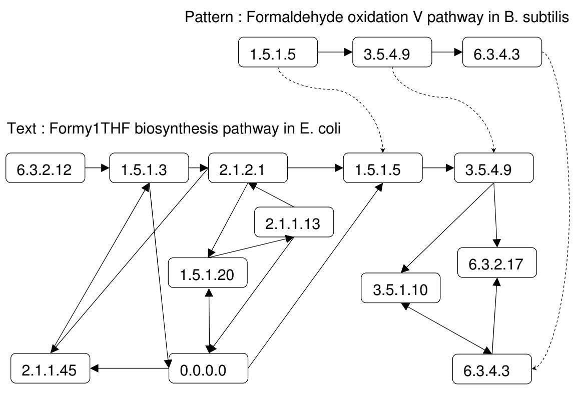


Figure 5.3. Mapping of formaldehyde oxidation V pathway in *B. subtilis* to formy1THF biosynthesis pathway in *E. coli* ($p < 0.01$). The node with upper part and lower part represents a vertex-to-vertex mapping. The upper part represents the query enzyme and the lower part represents the text enzyme. The node with dashed box represents a gap.

5.8.5 Phylogenetic Validation.

One can measure similarity between species based on the number of conserved pathways. The largest amount of conserved pathways is found between *B. subtilis* and *T. thermophilus* – two species-to-species mappings have in total 183 statistically significant pairs of pathways. The next closest two species are *E. coli* and *B. subtilis* which have 126 statistically significant pairs of pathways. This agrees with fact that *B. subtilis*, *T. thermophilus*, and *E. coli* are prokaryote and *S. cerevisiae* is a eucaryote.

5.9 Conclusions

The proposed alignment approach allows to map different enzymes of the pattern pathway into a single enzyme of a text network. We have also defined a novel scoring scheme for computing similarity between enzymes based on their EC notation.

We have formulated the graph-theoretical problem corresponding to finding optimal network alignment from the pattern network to a text network. We give an efficient dynamic-programming method for exactly solving this problem when the pattern is polytree an the text is an arbitrary network.

We have applied our mapping tool pairwise mapping of all pathways for four organisms (*E. coli*, *S. cerevisiae*, *B. subtilis* and *T. thermophilus* species) representing main different lineages and found a reasonably large set of statistically significant pathway similarities.

We report 24 pathways that are conserved across all 4 species as well 18 more pathways that are conserved across at least three of these species. We show that our mapping tool can be used for identification of potential pathway holes as well resolving enzyme notation ambiguities in existing pathway descriptions.

CHAPTER 6
NETWORK ALIGNMENTS FOR GENERAL PATTERNS

We further extend our approach to cyclic patterns. First considering that every connected subgraph has a spanning tree, an intuitive idea is to enumerate all possible subgraph which covers all vertices, to run our previous algorithm on their spanning trees, and finally to recover the connection of edges in pattern which disappear in the spanning trees and search for global optimal alignment. If no deletions in pattern are taken into account, an easier approach is to find minimum feedback vertex set (FVS) which removal cuts cycles out, iteratively to "fix" the vertices in FVS to text vertices, and finally to obtain the global optimal alignment. Further if pattern vertex deletions happen only on vertices belonging to non feedback vertex set, the previous algorithm can be directly employed. If pattern vertex deletions are allowed on FVS vertices, the possibility of possible subgraphs generated by bypass deletions is exponential and a series of data reductions of the general pattern are required. The idea of data reduction is to quickly presolve those parts of the input data that are relatively easy to cope with, shrinking it to those parts that form the really hard core of the problem [30]. In our case, the presolved parts are the subgraphs without deletions of vertices in FVS. The other subgraphs can be handled by the reduction of a series of the cases same as the one of the presolved parts. The resulted algorithm is a fixed-parameter combinatorial algorithm which increases the runtime by the factor of $V_T^{|FVS|^2}$.

Besides, for improving the performance and at the same time guaranteeing to find optimal solutions, we design and construct a tree decomposition structure of the pattern and proposed a first algorithm which align the treelike-property structure with the text and increases run time by the factor of $V_T^{c|TD|}$ (c is constant and $|TD|$ is the treewidth of the pattern (Especially the treewidth of metabolic network is 2).).

In this chapter, I first present algorithms based on feedback vertex set and then present tree decomposition and its application in network alignment.

6.1 Handling cycles in patterns by feedback vertex set

6.1.1 Handling cycles in patterns without pattern vertex deletions

The dynamic programming algorithm for multi-source patterns heavily relies on the existence of sorting of P such that for any vertex v all children cannot communicate with each other except through v. In order to have the same property for patterns with cycles, we "fix" the images of the vertices from $F(P)$ in the text T, called feedback vertex set or cycle cut vertex set, i.e., we assume that for each $v \in F(P)$ we know its image $f(v) \in V_T$.

Searching for the minimum cost homomorphism $f : P \to T$ among only mappings that preserve mapping pairs $(v, f(v)), v \in F(P)$, can be done efficiently. Indeed, let K be a connected component of $P \setminus F(P)$ and let K' be the connected component of $K \cup F(P)$ containing K. The vertices of K' are sorted in such a DFS order that feedback vertices are leaves. We then run our algorithm from the previous section with the assumption that the text images of feedback vertices are fixed.

In order to find the overall optimal homo-homeo morphism we should repeat the above procedure with all possible fixed mappings of the feedback vertices. The total number of such mappings is $O(V_T^{|F(P)|})$ and can be very large if $|F(P)|$ is large.

We further improve the runtime of our algorithm by reduction to the minimum weighted feedback set problem. The text T usually contains very few vertices corresponding to enzymes that have EC annotation similar to EC annotation of v. Let $t(v)$ be the number of possible text images of a given pattern vertex v. Then the total number of all possible feedback set mappings is $O(\prod_{v \in F(P)} t(v))$ rather than $O(V_T^{|F(P)|})$. In order to minimize that amount we can minimize its logarithm $\sum_{v \in F(P)} \log t(v))$. Finally, we need to find the minimum weight feedback set of the pattern P where the weight of each vertex v is $\log t(v)$. This problem is NP-complete and we have implemented the 2-approximation algorithm [6].

6.1.2 Calculation of the penalties for pattern vertex deletions

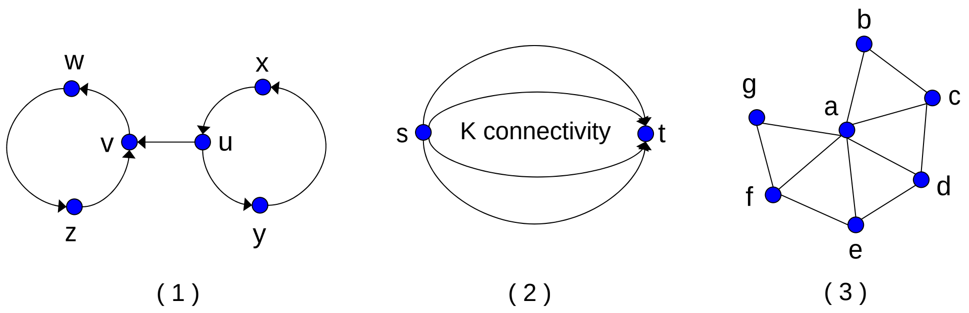


Figure 6.1. Examples for the discussion about pattern vertex deletion. (1) An example which specifies pattern vertex deletions so that induced subgraph can be generated; (2) An example to show induced subgraph reserves the k connectivity of two vertices in original graph. (3) An example which demonstrates general pattern vertex deletion leads to a combination problem.

The specified *strong deletion* corresponds to the operation of deleting subgraphs of the pattern and obtaining a vertex-induced subgraph. A vertex-induced subgraph P', sometimes simply called an "induced subgraph", is a subset of the vertices of a graph P together with any edges whose endpoints are both in this subset. Namely, if a reachable path in the graph G runs across any two vertices in P', the path should be reserved in P'. Let us take cases in Fig. 6.1 as examples. These examples can be used to represent classes of connectivity patterns which are generated by recursively deleting degree-1 vertices. As shown in Fig. 6.1.1, under the specified definition of strong deletion, the bridge $u \to v$ between two connected subcomponents can not be deleted if either or both of the subcomponents are not deleted; vertices in the connected subcomponent such as $v \to w \to z \to v$ except the articulation point v can be either totally deleted or totally reserved. As shown in Fig. 6.1.2, once two vertices s and t stay in the induced subgraph, the k^{th} connectivity between them in original graph should be kept.

The specified *bypass deletion* of a vertex of degree 2 supports homeomorphism, i.e., it allows the replacement of a path with a single edge. If the pattern is a directed graph, then a vertex v can be bypassed only if it belongs to a directed path $a \to v \to b$, and consequently the incoming and outgoing edges are replaced by a single edge $a \to b$. Both types of deletions can be applied recursively and together with each other.

The two types of pattern vertex deletions and their combination avoid the occurrence of arbitrary pattern vertex deletions which may leads to combination problems and exponential runtime solutions in a graph with multiple cycles (see Fig. 6.1.3 as an example).

6.1.3 An combinatorial algorithm for general network alignments

Denote FVS as feedback vertex set. Denote $deg(v)$ as the degree of vertex v. Denote $inEdges(v)$ and $outEdges(v)$ as the set of incoming and outgoing edges of vertex v, respectively. If v is a degree-2 vertex, denote $inEdge(v)$ and $outEdge(v)$ as the incoming and outgoing edges of vertex v. As in chapter 5, a *bypass deletion* of a vertex v corresponds to mapping v into **b**. A *strong deletion* of a vertex v corresponds to mapping v into **d**.

Let us call HH the algorithm of optimal network alignment from a polytree to an arbitrary graph (see chapter 5). Let us call HH' the following algorithm for general network alignments (see its problem formulation in [18, 17]).

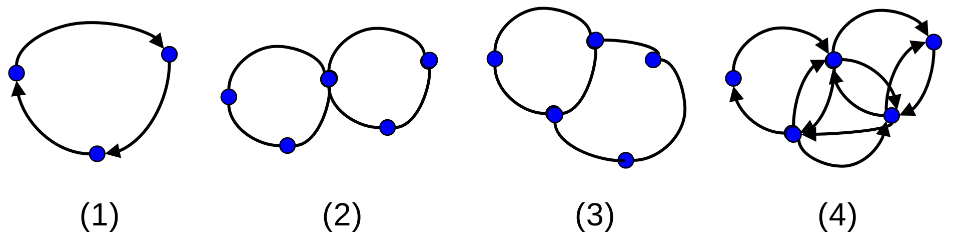


Figure 6.2. Cycle patterns for case 1, 2, 3, and 4 respectively.

Input: graph $P =< V_P, E_P >$ and $T =< V_T, E_T >$ respectively as pattern and text.

Case 1: P is a unicyclic graph (when directions are disregarded):

```
    Let $C$ be vertices in the cycle
    Subcase 1.1: Only strong deletion happens on the vertex in $FVS$
        for each $v \in C$
            $P' \leftarrow P \setminus \{v\} \setminus (descendants(v) \setminus C)$
            align $P'$ with $T$ by $HH$
        $A_{1.1} \leftarrow$ minimum cost alignment

    Subcase 1.2: Bypass deletion happens on the vertex in $FVS$
    if $\exists v \in C$ & $deg(v) == 2$
        for each $v \in V_P$
            align $C \cup \{e = (inEdge(v), outEdge(v))\} \setminus \{v\}$ with $T$ by applying case 1.2
        $A_{1.2} \leftarrow$ minimum cost alignment

    Subcase 1.3: No deletion on $FVS$ vertex
    if $\exists v \in C$ & $deg(v) > 2$
        for each $u \in V_T$
            "fix" $v \rightarrow u$
            do alignment by $HH$
        $A_{1.3} \leftarrow$ minimum cost alignment
    $A_1 \leftarrow \min A_{1.1}, A_{1.2}, A_{1.3}$

case 2: $P$ is not a uni-cycle graph & $|FVS| = 1$ & $v \in FVS$
    Subcase 2.1: $v \rightarrow \mathbf{d}$
    for each disjoint component $P'$ of $P \setminus \{v\}$
        align $P'$ with $T$ by $HH$
    $A_{2.1} \leftarrow$ minimum cost alignment
    Subcase 2.2: $v \rightarrow \mathbf{b}$
    for each pair of $e1 \in inEdges(v)$ and $e2 \in outEdges(v)$
        $P' \leftarrow$ keep the branch of $e1$ and $e2$, delete the other branches, and bypass delete $v$
        align $P'$ with $T$ by applying case 1
    $A_{2.2} \leftarrow$ minimum cost alignment

    Subcase 2.3: No deletion on $v$
    for each $u \in V_T$
        "fix" $v$ to $u$
        do alignment by $HH$
    $A_{2.3} \leftarrow$ minimum cost alignment
    $A_2 \leftarrow \min A_{2.1}, A_{2.2}, A_{2.3}$

case 3: $|FVS| = f$ & $P$ is bi-connected
    for each permutation $l = (v_{f_1}, v_{f_2}, ..., v_{f_i})$ of vertices in $FVS$
        for each $v_{f_j} \in l$, $j$ is from 1 to $i$

            Subcase 3.1: $v_{f_j} \rightarrow \mathbf{d}$
              for each disjoint component $P'$ of $P \setminus \{v_{f_j}\}$
                align $P'$ with $T$ by applying $HH'$
              $A_{3.1} \leftarrow$ minimum cost alignment

            Subcase 3.2: $v_{f_j} \rightarrow \mathbf{b}$
              for each pair of $e1 \in inEdges(v)$ and $e2 \in outEdges(v)$
                $P' \leftarrow$ keep the branch of $e1$ and $e2$, delete the other branches, and bypass delete $v_{f_j}$
                align $P'$ with $T$ by applying $HH'$
              $A_{3.2} \leftarrow$ minimum cost alignment
            Subcase 3.3: no deletion on $v_{f_j}$
              for each $u \in V_T$
                "fix" $v_{f_j} \rightarrow u$
                do alignment by applying $HH'$
              $A_{3.3} \leftarrow$ minimum cost alignment
            $A_{3_l} \leftarrow \min A_{3.1}, A_{3.2}, A_{3.3}$
        $A_3 \leftarrow \min A_{3_l}$

case 4: $|FVS| = f$ & $P$ is not bi-connected
    find the set of articulation points $S$
    for each permutation $l = (v_{s_1}, v_{s_2}, ..., v_{s_i})$ of vertices in $S$
        for each $v_{s_j} \in l$, $j$ is from 1 to $i$
            do same thing as case 3.1, 3.2, and 3.3 and save results to $A_{4.1}, A_{4.2}, A_{4.3}$
            $A_{4_l} \leftarrow \min A_{4.1}, A_{4.2}, A_{4.3}$
    $A_4 \leftarrow \min A_{4_l}$
```

Computational complexity analysis.

In chapter 5, the runtime for HH algorithm is $O(|V_P| \times (|E_T| + |V_T| log|V_T|) + |E_P||V_T|)$.

For case 1 in HH', obviously it will be increased by the factor $O(V_P^2)$. For case 2, the number of pairs of incoming edges and outgoing edges is E_P^2. So the runtime for the case is increased by the factor $O(V_P^2 \times E_P^2)$.

For case 3.2, every feedback vertex set vertex v has $|inEdges(v) \times outEdges(v)|$ possibilities to be bypass deleted. We specify the mapping as $v \rightarrow \mathbf{b}_i$, $i \in [1, |inEdges(v) \times outEdges(v)|]$. However, every vertex may be mapped to a vertex in text T, $\mathbf{d}$, or $\mathbf{b}_i$.

For every iteration, the feedback vertex size is reduced by at least 1. We use FVS_i to represent the feedback vertex set in iteration i and $v_{f_{i,j}}$ to represent the vertex in FVS_i . So the total runtime in the case is increased by the factor

$$O(\Pi_i(V_T + 1 + max_j(inDegree(v_{f_{i,j}}) \times outDegree(v_{f_{i,j}})))^{|FVS_i|}$$

$$= O((V_T + 1 + max(inDegree(v_P) \times outDegree(v_P)))^{|FVS|^2})$$

For case 3 or case 4, the upbound runtime increase factor is

$$O((V_T + 1 + max(inDegree(v_P) \times outDegree(v_P)))^{|FVS|^2})$$

As a consequence, the total runtime of the algorithm is

$$O((V_T+1+max(inDegree(v_p) \times outDegree(v_P)))^{|FVS|^2} \times |V_P| \times (|E_T|+|V_T|log|V_T|)+|E_P||V_T|))$$

6.1.4 A reduction algorithm for general network alignments

Let $G =< V, E >$ be a directed graph. A neighbor of $u \in V$ is a vertex $v \in V$ which is adjacent to u by an edge in E. Its degree $d(u)$ is the number of edges adjacent to u in G. A vertex in G of degree 1 is called a *leaf.* A vertex of degree 2 is called a *bypass vertex.* A *cycle* is a closed alternating sequence of adjacent vertices and edges that contain no repeated edges. A simple cycle is a cycle that contains no repeated vertices. Two cycles in G are *independent* if their vertex sets are disjoint; they are *weakly independent* if their edges sets are disjoint. A graph is called a *self-loop* if it contains only a vertex and an edge adjacent to itself.

A *feedback vertex set* of G is a subset of vertices $F \subseteq V$ such that each cycle in G pass through at least one vertex in F. The removal of the vertices in F and all their edges makes G cycle-free. A *minimum feedback vertex set* of a graph G is a feedback vertex set F_{min} of the minimum size.

A *feedback arc set* of G is a subset of edges $F_E \subseteq E$ such that each cycle in G contains at least one edge in F_E. The removal of the edges in F makes G cycle-free. A *minimum feedback arc set* of a graph G is a feedback arc set $F_{E_{min}}$ of the minimum size.

The cyclomatic number $r(G)$ of a graph G is the size of the minimum feedback arc set. Let F-*cycle* be a graph with exactly F cycles such that any pair of them are pairwise vertex-disjoint (See Fig. 6.3 as an example). Furthermore we denote by $\delta_G(v)$ the decrease in the cyclomatic number of the graph on removing vertex v.

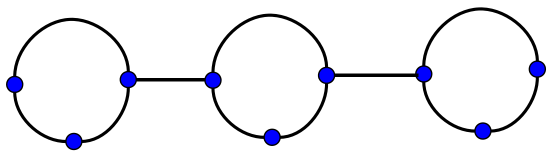


Figure 6.3. An F-cycle sample.

First we define a procedure to reduce G to G'.

```
Algorithm 6.1.4.1
Input: graph $G$;
Output: reduction $G'$ of $G$

$H \leftarrow G$
while $v$ is a leaf in $H$ do
    delete $v$ and its adjacent edges from $H$
while $v$ is a bypass vertex in $H$ without a self-loop do
    add an edge to connect the two neighbors of $v$
    remove $v$ and its edges
$G' \leftarrow H$.
```

Lemma 1. The reduction doesnot reduce the number of minimum feedback vertex set. Namely, let G be a graph and G' be its reduction by the above procedure. Then, every feedback vertex set of G' is also a vertex feedback set of G.

Proof. Let H_i be the values of H in the beginning of i_th iteration. Let H_{i+1} be the values of H in the end of i_th iteration. Clearly the replacement of a bypass vertex and its edges with an edge connected the vertex's neighbors keeps cycles.

Now we list a greedy algorithm which achieves an approximation guarantee of factor 2 for the feedback vertex set problem.

```
Algorithm 6.1.4.2
Input: graph $G$;
Output: a feedback vertex set V;
$H \leftarrow G$, $i \leftarrow 0$
while $u \in V_H$ do
    $w'(u) \leftarrow 1$
while $H$ is cyclic do
    $c \leftarrow min_{u \in V_H}\{\frac{w'(u)}{\delta_{H(u)}}\}$
    $G_i \leftarrow H$
    while $u \in V_H$ do
        $t_i(u) \leftarrow c \times \delta_{G_i}(u)$, $w'(u) \leftarrow w'(u) - t_i(u)$
    $H \leftarrow$ the subgraph of $G_i$ induced by vertices $u$ with $w'(u) > 0$
    $i \leftarrow i + 1$
$k \leftarrow i$, $G_k \leftarrow H$


$F_k \leftarrow \emptyset$
for $i \in [k \dots 1]$ do
    $F_{i-1} \leftarrow F_i \cup (V_{i-1} - V_i)$
$F \leftarrow F_0$.
```

Lemma 2. Algorithm 6.1.4.2 achieves an approximation guarantee of factor 2 for the feedback vertex set problem.

Proof. see vazirani's textbook.

A faster reduction algorithm.

The feedback vertex set (FVS) of a graph G is a set of vertices whose deletion makes graph acyclic. Let $f(G)$ denote the size of the minimum FVS of G. Denote $deg(v)$ as the degree of a vertex v. Denote $inEdges(v)$ and $outEdges(v)$ as the set of incoming and outgoing edges of v, respectively. If v is a degree-2 vertex, denote $inEdge(v)$ and $outEdge(v)$ as the incoming and outgoing edges of v. As in [17], a *bypass deletion* of v corresponds to mapping v into **b**. A *strong deletion* of v corresponds to mapping v into **d**.

Cases:

Case 1. No strong deletions and no bypass deletions are allowed. In polytrees we run DP algo A_0. For graphs with cycles, choose an arbitrary FVS $F \subseteq V_P$ and map each its vertex into each vertex of T, then obtain the optimal mapping of each resulted connected component using polytree DP algo A_0. The runtime is $O({V_T}^F rt(A))$ [19].

Case 2. No strong deletions are allowed. In polytrees we run DP algorithm A_b. For graphs with cycles, the same algorithm as in Case 1, but we should make sure that no vertex in F is bypass deleted. So we choose FVS with each vertex having in- or outdegree at least 2 and the runtime is $O({V_T}^F rt(A_b))$. Such FVS always exists unless P is a loop, i.e., a directed cycle in which all vertices have in- and outdegree equal to 1. In this case, at least one vertex will not be bypass deleted and for each vertex v in this cycle C we map v into each vertex of T, run polytree DP algorithm A_b, and choose the best mapping. The total runtime is $C \times rt(A_b)$. Thus the runtime is $O(({V_T}^F + V_P)rt(A_b))$.

Case 3. Both strong deletions and bypass deletions are allowed. In polytrees we run DP algorithm A_{bd}. For graphs with cycles, the same algorithm as in Case 2 works if no vertex in FVS is allowed to be bypass deleted. The case of a single loop C as a pattern is handled the same way with the runtime $C \times rt(A_{bd})$ and the runtime would be $O(({V_T}^F + V_P)rt(A_{bd}))$.

Unfortunately, even though a vertex v cannot be originally bypass deleted since it has in- or out-degrees higher than 1, the strong deletion of some of its neighbors can make v eligible for bypass deletion.

If at least one vertex $v \in F$ is not bypass deleted, then this vertex should either be mapped into T or strong deleted and the rest of the pattern graph has FVS reduced by 1.

Now consider the case when *all* vertices in F are bypass deleted. The number of ways how a vertex v can be bypass deleted is equal to $indeg(v)outdeg(v)$. Each such bypass deletion requires strong deletion of $indeg(v) + outdeg(v) - 2$ neighbors. Let f be the number of all possible combinations of bypass deletions of all FVS vertices, i.e.,

$$S = \prod_{v \in F} indeg(v)outdeg(v) < {V_P}^{2F}$$

The cyclomatic number of a graph G is the minimum number r of edges whose removal makes G cycle-free. Let F*-cycle* be a graph with exactly F cycles such that any pair of them are pairwise vertex-disjoint.

Lemma 1. A graph G is an F-cycle if and only if the size of minimum feedback vertex set and cyclomatic numbers are equal F, $f(G) = r(G) = F$. If G is not an F-cycle, then a greedy algorithm outputs a FVS of size at most $F - 1$.

Proof. Let G be an F-cycle. Since all cycles are vertex disjoint they are also edge disjoint and it necessary and sufficient to delete F vertices or edges to destroy all F cycles.

Now let $f(G) = r(G) = F$. There exists an edge set $X = \{e_1, \ldots, e_F\}$, such that $G - X$ is acyclic. On the contrary, assume that G has a vertex v belonging to two different cycles, then X should contain two different edges e_i and e_j that break them. These two edges can be replaced in X with edges e'_i and e'_j incident to v. Then $f(G) \leq F - 1$, since we can choose a FVS consisting of v and one endpoint from each edge in $X - e_i - e_j$.

If G is not an F-cycle, then there should be a vertex that belong to at least two cycles. A greedy algorithm will find such vertex and remove from G. Therefore, it will output a FVS of size at most $F-1$. □

We can handle F-cycle with the following algorithm B_{bd}. We find the leaf cycle L (the one which is connected to at most one other cycle) and its stem (v,u) ($v \in L$ and (v,u) is on the unique path connecting L with the rest of the F-cycle). We recursively map v. There are 4 cases:

1. If v is mapped into T or strong deleted then we need also to map the remaining $(F-1)$-cycle graph.
2. If v is bypass deleted and u is not strongly deleted, then there are $indeg(v)$ such possibilities each supplemented with mapping of the remaining $(F-1)$-cycle graph.
3. If v is bypass deleted and u is strongly deleted and one of the vertices adjacent to v in L is deleted.
4. If v is bypass deleted and u is strongly deleted and vertices adjacent to v in L are not deleted.

Let R_F be the runtime for finding optimal mapping of F-cycle. Then

$$R_F \leq (V_T+1)R_{F-1} + indeg(v)R_{F-1} + indeg(v)outdeg(v)R_0 + LR_0,$$

where $R_0 = rt(A_{bd})$. Thus, the runtime of the algorithm B_{bd} is $rt(B_{bd}) = O((2V_T)^F rt(A_{bd}))$.

Lemma 2. Let P' be the graph after strong deletion of $\sum_{v\in F}(indeg(v)+outdeg(v)-2)$ vertices and bypass deletion of F FVS vertices. Then either we can find with a greedy algorithm FVS of P' of size at most $F-1$ or P' is an F-cycle graph.

Proof. Since all F vertices are bypass deleted in P', removal of the corresponding F edges makes P' acyclic. Therefore, $r(P') \leq F$. If P' is not a F-cycle, then by Lemma 1 the greedy algorithm will find a FVS of size at most $F-1$. □

Thus we obtain the following recursion for the runtime $RT(P,F,T)$ for the optimal mapping of the pattern P with P vertices and VFS of size F into the text T with T vertices:

$$\begin{aligned} RT(P,F,T) &\leq F \cdot T \cdot RT(P,F-1,T) + S \cdot RT(P,F-1,T) \\ &\leq (FV_T + S) \cdot RT(P,F-1,T) \\ &\leq (FV_T + V_P^{2F}) \cdot RT(P,F-1,T) \end{aligned}$$

At any point, if we obtain an F-cycle graph.

This recursion implies that

$$RT(P,F,T) \leq ((FV_T + V_P^{2F})^F + rt(B_{bd}))rt(A_{bd})$$

and we have proved the following

Theorem. The optimal network alignment problem can be solved in time $O(((FV_T + V_P^{2F})^F + rt(B_{bd}))rt(A_{bd}))$. When the FVS size is bounded this problem becomes tractable.

6.2 A polynomial-time algorithm for series-parallel patterns

6.2.1 Introduction

Metabolism is a vital cellular process whose understanding is critical to human disease studies and drug discovery. The accumulation of high-throughput genomic, proteomic and metabolical data allows for increasingly accurate modeling and reconstruction of metabolic networks. Comparison among the reconstructed networks can catch model inconsistencies and infer missing elements. With the growth of identified metabolic networks, computational tools are necessary for the comparison. Network alignment is convenient for comparing and exploring metabolic networks – it can be used for predicting unknown or alternative pathways and pathway holes as well as resolving ambiguities and finding inconsistencies in existing pathway descriptions.

A metabolic pathway/network can be represented as a directed graph in which vertices are enzymes. Each of its edges connects enzymes catalyzing consecutive reactions. Pinter et. al. [43] formulated the network alignment as a labeled subtree homeomorphism problem – given a vertex labeled pattern tree P (representing an unknown pathway) and text graph T (representing a known pathway), find the minimum cost transformation of P into subtrees of T by edge subdividing with degree-two vertices. Pinter et. al. [44] gave an efficient algorithm for the case when T is a tree. Sze [61] allowed to delete pattern nodes and gave an efficient algorithm for path matching and an exponential algorithm for arbitrary pattern and text graphs.

Cheng et. al. [17] proposed an optimal alignment between a polytree pattern and text graphs allowing enzyme deletion and insertion as well as matching similar enzymes. Their algorithm is efficient for arbitrary text graphs and pattern graphs with a restricted cyclic structure. But for the pattern with feedback vertex set of size k, the runtime is exponentially bounded by k^2. Due to mimicking evolutionary machinery of gene duplication ([48]), similarly to [61] their algorithm allows to map different pattern enzymes into the same text enzyme.

Dost et. al. [22] formulated the problem as optimal homomorphism problem. Authors employed color coding technique to mapping query tree to arbitrary graph by two steps. The first step is to randomly do coloration of pattern Q which has 2^k possibilities (k is the size of the color set). The second step is to compare the randomly generated colored graph with N. For any general query including multiple cycles, authors designed a tree decomposition based algorithm which did not be implemented. The algorithm has the runtime $O(2^k \times |V_T|^{w+1})$, in which w is the treewidth of the query graph.

Furthermore, Bruckner et. al. [12] proposed an approach of network querying that does not rely on knowledge of the query topology. The problem is formulated to search a colored graph for connected subgraphs whose vertices have distinct given colors. Authors provided fixed-parameter algorithms that are based on the color-coding paradigm and dynamic programming (DP). In addition, authors provided an integer programming (ILP) formulation of the problem. The methods extended the work of QNet and can handle edge weights, insertions of network vertices (that do not match any query protein), and deletions of query nodes. The algorithm runs in $O(3^k \times |E_T|)$ time for handling the alignment without insertions and deletions of vertices.

However, the color-coding technique [4] can find an optimal alignment with the probability of $\frac{k!}{k^k} = e^{-k}$, which cannot guarantee to obtain such an optimal result at all times. As a result of the homomorphism based formulation, insertion of query nodes is not necessary, which are needed for the alignment to search for conserved patterns.

Here, for finding the best matching pair consisting of a subgraph in a given pattern and a subgraph in a given text (both are represented by an arbitrary network) when both insertions and deletions are allowed on any path, we employed the technique of tree decomposition and proposed a fixed-parameter algorithm which is exponentially bounded by the treewidth of the pattern. We have applied our algorithm to do experiments in the cases of self-alignment, deletions, and insertions. We also provided the MetNetAligner web service tool which relies on the algorithm for metabolic network alignments.

6.2.2 The network alignment problem

Network alignment is a graph-comparison problem where vertices in graph are compared and graph topology is also compared. Its goal is to determine if or not the labels and network topology, or part of them, of two graphs are similar. It allows the mismatches of vertices of two graphs and deletions or insertions of vertices in graphs.

We distinguish two kinds of vertex deletions: (i) bypass deletion corresponding to the replacement of a few consecutively acting enzymes with a single multifunctional enzyme or enzyme using an alternative catalysis and (ii) strong deletion symbolizing the matching of a proper connected subgraph of the pattern network. Thus, a *bypass deletion* of a vertex v with a single incoming and a single outgoing edge is the replacing of a u-v-w-path with a single (u,w)-edge formally represented as mapping v into $\mathbf{b}$. A *strong deletion* of a vertex v is removing of arbitrary vertex v together with all its incoming and outgoing edges which is formally represented as mapping of v into $\mathbf{d}$.

Let graphs $P = (V_P, E_P)$ and $T = (V_T, E_T)$ represent a pattern and text metabolic network, respectively. Let $f : P \to T$ map every vertex in V_P to $V_T \cup \{\mathbf{b}, \mathbf{d}\}$. A pair of pattern vertices $u, v \in V_P$ is called *contracted* if u and v are mapped into two different text vertices and either $(u,v) \in V_P$ or there exists a u-v-path in P whose all intermediate vertices are bypass deleted. Let E_P^f be the set of all vertex pairs in P contracted by f (note that $E_P \subseteq E_P^f$). The mapping f is called a *network alignment* if for each contracted pair $(u,v) \in E_P^f$ there exists a $f(u)$-$f(v)$-path in the text T. The number of insertions, i.e., the minimum number of intermediate vertices in such path is denoted $\sigma(f(u), f(v))$.

The alignment should be penalized (i) for mismatches between aligned enzymes, (ii) for strong deletions, (iii) for bypass deletions, and (iv) for insertions. Thus we obtain the following cost function.

$$cost(f) = \sum_{u \in V_P} \Delta(u, f(u)) + \lambda \sum_{(u,v) \in E_P^f} \sigma(f(u), f(v)),$$

where $\Delta(u, f(u))$ is the penalty of the mismatch between enzymes corresponding to pattern vertex u and text vertex $f(u)$, $\Delta(u, \mathbf{d})$ and $\Delta(u, \mathbf{b})$ are penalties for strong and bypass deletions, respectively, and λ is the penalty for a single enzyme insertion.

We proposed an enzyme-to-enzyme dissimilarity score Δ based on 4-level EC encoding d1:d2:d3:d4, where d4 is the number of enzyme in d3-class which is subclass of d2-class which is subclass of d1-class. The numeric values are usually positive integers. They are represented as 0 or - only if the corresponding subclass is unknown. If d1's or d2's of two enzymes are different and non-zero, then $\Delta = \infty$, otherwise, if d3's are different and non-zero, then $\Delta = 10$, otherwise if d4's are different and non-zero, then $\Delta = 1$, and, otherwise, $\Delta = 0$ (see [19]). Notice that if two enzymes are different in the corresponding position but one of them is 0, then we do not increase Δ. This is because we do not penalize the lack of information.

Given two metabolic networks P and T, the problem is to find the optimal (minimum-cost) network alignment from P to T.

6.2.3 Methods

6.2.3.1 The decomposition tree of pattern

Numerous graph problems can be solved efficiently for trees because of the following property: each non-leaf node of a tree splits it into multiple connected components, so we can (a) **divide** the tree into those components, (b) make all possible assumptions about the separating node, (c) **conquer** the components, i.e. solve the subproblems specified by the component and the assumption about the separating node, (d) combine the matching subproblems (i.e. having consistent assumptions) into total solutions.

Consider a local decomposition algorithm for the solution with a tree-like structure, i.e. to find the set of the neighborhoods of different variables so that one variable can belong to two neighborhoods only and the graph of intersections of these neighborhoods is a tree. It is clear that such a structure is a tree-decomposition and can be obtained with the aid of known tree decomposition algorithms. These tree decomposition alogorithms aim to merging variables such that the meta-graph is a tree of meta-nodes. The tools can help detect trees and obtain the treewidth, a measure of the tree-likeness of the graph. The notions of treewidth and tree decomposition were introduced by Robertson and Seymour in their seminal paper [51] on graph minors [46].

A tree decomposition of graph (V, E) is a tree (I, F, X) where each tree node $i \in I$ has a set $X_i \subset V | X_i \in X$, so that

for each $v \in V$ the set $\{i \in I : \ v \in X_i\}$ is connected in (I, F, X), and

for each $e \in E$ there exists $i \in I$ such that $e \in X_i$.

The width of decomposition (I, F, X) is $\max_{i \in I} |X_i| - 1$, and the tree width of (V, E) is the minimum width of a tree decomposition of (V, E).

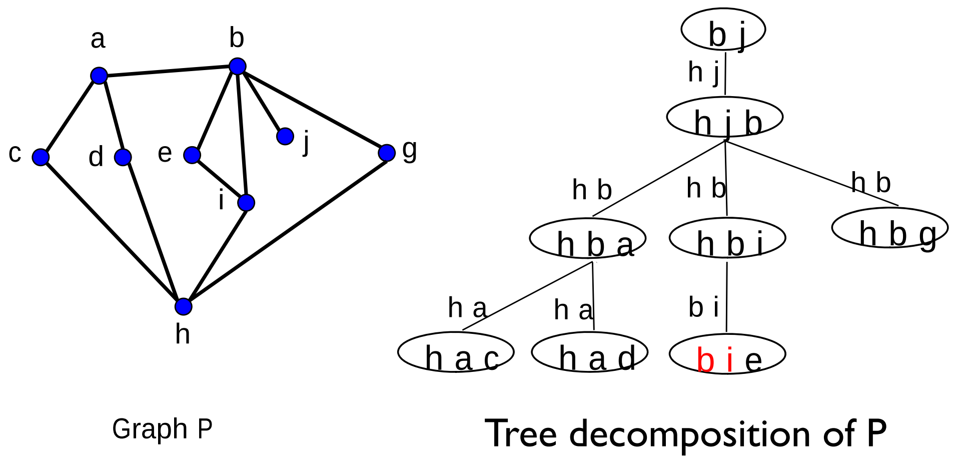


Figure 6.4. A sample graph P and its decomposition tree. The red two letters in the tree node represent the true connection between separators.

With Dr. Piotr Berman's help, we designed and implemented a hierarchical decomposition of the underlying graph. The decomposition is constructed by first reducing the graph by choosing a series of nested vertex separators of subgraphs into smaller and smaller pieces, until all pieces contain only one vertex. The vertex separators are the set of vertices which removal disconnects the resulted subgraph. Every path between the subgraphs passes through some member of the separators. The vertex separator is also called as vertex cut. The subgraphs are not intersecting with each other. The process of hierarchical decomposition constructs a tree, which covers all vertices in the underlying graph. In the tree, every node represents a subgraph and its children represent the different connected components after cutting the separators of the parent. We label the parent by the set of the separators and an owner which is reachable to all vertices in the separator set. The owner belongs to the separator set of every child. The number of child tree nodes is less than the size of the separator set and every vertex in the set belongs to at most one child.

As a result of the decomposition algorithm, the edges of the tree define a "nice" collection of separators: consider a tree edge $e = \{i, j\}$. $(I, F - \{e\})$ has connected components I_i, I_j.

We can define the following subsets of V: $V_i = \bigcup_{\ell \in I_i} X_\ell$, analogous V_j and $S_e = X_i \cap X_j$. It is easy to see that (a) $V = V_i \cup V_j$, (b) there are no edges from $V_i - S - e$ to $V_j - S_e$, (c) $|S_e| \leq$ the width of (I, F, X).

The pseudo-code is listed below (An example and its tree is shown in Fig. 6.4):

Let v be a vertex in undirected graph G. Denote neighbors(v) to be the set of neighbors of v. Denote deg(v) to be the number of the degree of v. Let $Q = \{v|deg(v) = 2\}$ be a queue used to collect degree-2 vertices.

```
clone graph $G$ and save as $G' =< V', E' >$
insert $\{v|v \in V' \& deg(v) = 2\}$ into $Q$
initialize tree $TD$
**while** $Q$ is not empty
    **while** $Q$ is not empty
        delete from $Q$ an item $u$
        **if** $u$ is an articulation point and $neighbors(u) > 1$
            create a chain of tree node(s) $n_i|i \in [0..|neighbors(u)|]$ and insert into $TD$ the chain
        **else**
            create tree node $n$ with $u\&neighbors(u)$ and insert into $TD$ the node
        delete $u$ from $G'$
        if $|V'| > 3$
            insert $\{v|v \in V' \& deg(v) = 2\}$ into $Q$
    if $|V'| > 3$
        insert $\{v|v \in V' \& deg(v) = 1\}$ into $Q$
normalize tree $TD$
```

6.2.3.2 Series-Parallel graph

The associated decomposition tree represent the tree-like property of its corresponding graph. Its treewidth is the size of its largest separates minus one. A k-connected graph is the graph in which no such a set of $k-1$ vertices exists whose removal disconnects the graph. k-connected graph has the treewidth k. The treewidth indicates how close it is to being a simple tree.

Many important but quite hard graph problems can be solved efficiently on graphs of bounded treewidth [10, 54, 24, 8, 9, 57].

The evolution of metabolic networks is characterized by gain and loss of reactions (or enzymes) connecting two or more metabolites [20, 45]. The process of the gain leads to the occurrence of new alternative paths and new enzymes in existing paths, which corresponds to the insertion of new nodes or edges. The process of the loss corresponds to their deletions. The corresponding graphs are more probable to be series-parallel graphs which has bounded treewidth 2 [54, 10, 24]. These graphs can be obtained from edges by edge duplication and subdivision. The graph P in Fig. 6.4 is such a series-parallel graph.

If the graph has no K4 subdivision [1], we can claim that it is series-parallel. So by repeatedly doing the following operations, we can check if null graph is left: (i) deletion of a loop, (ii) deletion of a vertex of degree at most 1, (iii) deletion of a parallel edge, (iv) suppression of a vertex of degree two. Besides, there exist a linear-time algorithm to test whether an input graph is series-parallel [47].

Considering section 6.2.3.1, we can make the following observation. Suppose that node u is an articulation point of a graph (V, E), so we can decompose that graph into (V_0, E_0) and (V_1, E_1) where $V_0 \cap V_1 = \{u\}$. Assume that we have tree decompositions (I^0, F^0, X^0) and (I^1, F^1, X^1). Then we can combine these decompositions by adding an edge (i, j) such that $i \in I^0, j \in I^1$ and $u \in I^0 \cap I^1$. Thus a graph has tree width 2 iff every biconnected component has tree width 2. In turn, a biconnected graph has tree width 2 if and only if it is a series parallel graph. Also, one can convert series parallel decomposition into tree decomposition of width 2.

For the bounded tree width graph and its tree decomposition, given tree node i and its parent node j, we denote $Set(i)$ as the set of all vertices which are in x_i; we denote

$Cut(i)$ as its separators, which is equal to $Set(i) \cap Set(j)$ (These separator(s) belong to different subgraphs and its/their removal disconnect(s) the graph); we denote $Own(i)$ as the set of vertices only belonging to the subgraph represented by the tree node, which is equal to $Set(i) - Cut(i)$ or $Set[i] - Set[j]$. The separator is an articulation point if and only if $Cut(i) = 1$.

For the tree decomposition of a series parallel graph, it has the following characteristics. For every tree node i, $|Set(i)| \leq 3$, $|Own(i)| = 1$, and $|Cut(i)| \leq 2$. The separator is an articulation point if and only if $Cut(i) = 1$ and $Set(i) = 2$. We assume $Set[j] = \{u, v, w\}$, $Cut[j] = \{u, v\}$, i is the parent of j, and k is the child of j (if mentioned). We can observe that $Cut[k]$ contains w, i.e. it is $\{u, w\}$ or $\{v, w\}$ or $\{w\}$.

6.2.3.3 Algorithms

We have to discuss undirected cycles in directed graphs, so perhaps we can formalize the graph X as undirected, with direction function of edges, thus graph X has node set $V(X)$, edge set $E(X)$ that consists of 2-element subsets of $V(X)$ and for $e \in E(x)$ we define $dir(e)$, the node of e where the edge is directed.

Thus for $u \in V(X)$, $in(u) = \{v | \{u, v\} \in E \text{ and } dir(u, v) = u\}$ and $out(u) = \{v | \{u, v\} \in E \text{ and } dir(u, v) = v\}$.

However, we also need the notion of directed paths: a sequence $(u_0, \ldots, u_k)$ such that $u_i \in out(u_{i-1})$ for $i = 1, \ldots, k$ is a directed path from u_0 to u_k of length k; the least length of a directed path from u to v is $dist(u, v)$, if no such path exists, $dist(u, v) = \infty$.

We propose a fixed-parameter algorithm which finds optimal alignment of bounded tree width with arbitrary graph.For simplicity, we used series-parallel graph pattern which has tree width 2 to describe our algorithm.

In this section, I first describe graph preprocessing for both pattern and text; then I describe our dynamic programming.

- Preprocessing

Transitive closure of the text graph. In order to compute the cost of vertex insertion, it is necessary to know the number of hops for any shortest path in the text graph T. Essentially, the transitive closure of a directed graph contains an edge for every path in the graph.

Although finding single-source shortest paths in general graphs is slow, in our case it is sufficient to run breadth-first-search with runtime $O(|E_T| + |V_T|)$. Assuming that T is connected, i.e., $|E_T| \geq |V_T|$, we conclude that the total runtime of finding all shortest paths is $O(|V_T||E_T|)$. In the resulting transitive closure $T' = (V_T, E'_T)$ of the graph T, each edge $e \in E'_T$ is supplied with the number of hops $h(e)$ in the shortest path connecting its ends.

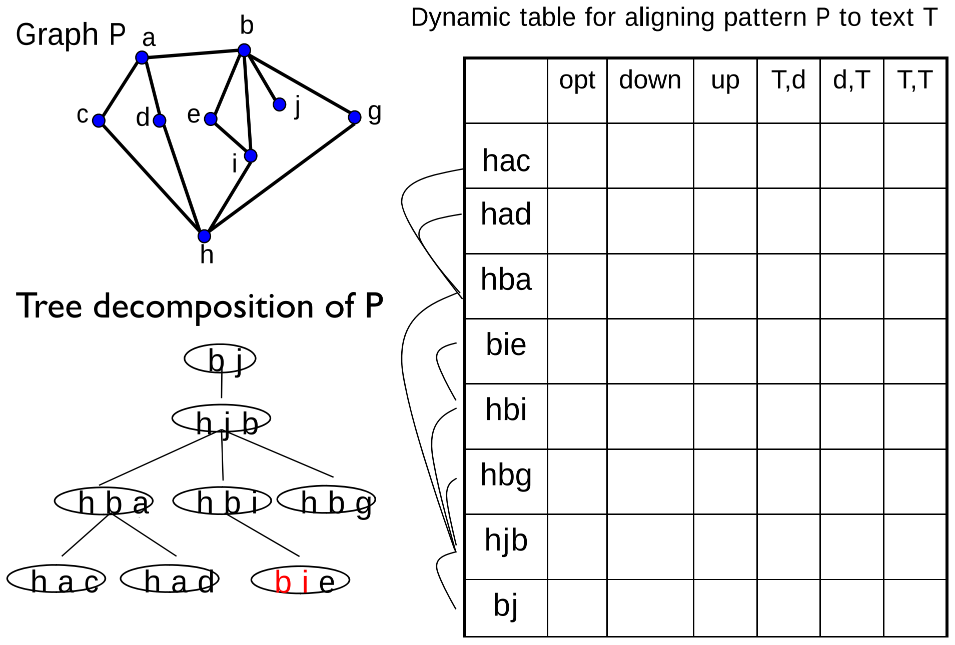
Pattern decomposition tree. First we build decomposition tree $TD =< I, F, X >$ of pattern P by the algorithm in section 6.2.3.1. Then we do the following normalization on the tree: If $Cut[k] \subset \{u, v\}$, we would make k a child of i.

Every node i in the decomposition tree represents an induced subgraph sub_i. We call its deletion as the downstream deletion and call the deletion of the remaining part(s) as the upstream deletion. By the post order traversal of the tree, we compute every subgraph's downstream and upstream deletion penalties, respectively denoted as $down(i)$ and $up(i)$ $(i \in I)$.

Alignment setting. We compute mismatching score $\Delta(v_P, v_T)$ for every pair of vertices v_P and v_T, one of which is from the pattern and the other of which is from the text.

We will further need a certain fixed order of nodes in TD as follows. Let $TD =< I, F, X >$ be the undirected decomposition tree obtained from P by disregarding edge directions. Let us choose an arbitrary vertex $r \in I$ as a root and run depth-first search (DFS) in TD from r. Let $\{i_1, \ldots, r = i_{|I|}\}$ be the order of the DFS traversal of I, correspondingly let $\{x_{i_1}, \ldots, r_x = x_{i_{|I|}}\}$ be the order of the visited subgraph, and every edge $e \in E_P$ exists in $x_{i_j}|j \in [1..|I|]$ or one of subgraphs.

- Dynamic programming approach



	opt	down	up	T,d	d,T	T,T
hac						
had						
hba						
bie						
hbi						
hbg						
hjb						
bj						

Figure 6.5. A sample graph P, its decomposition tree and its dynamic table (ignoring bypass deletions).

DP Table. Now we will describe our dynamic programming table $DT[1, \ldots, |I|][1, \ldots, 3 + 2 * |V_T| + |V_T|^2]$. In the table, each row of this table corresponds to a tree node of TD. The rows $\{i_1, \ldots, r = i_{|I|}\}$ of DT are sorted according to the DFS traversal of TD. Its entries in a row i give the minimum cost alignment of the subtree rooted by i under the mapping condition of the vertices in $Cut[i]$ and also decide the image of $Own[i]$ which make best contribution to the alignment.

Specifically, the first column corresponds to the local optimal alignment of pattern subgraph to the text T, which is denoted by $opt[i]$; the second and third column correspond to the downstream and upstream deletion respectively, denoted by $down[i]$ and $up[i]$ respectively; the next V_T columns correspond to the state that the first separator is mapped to arbitrary vertex in T and the second separator is mapped to $\mathbf{d}$; the following V_T columns correspond to the state that the first separator is mapped $\mathbf{d}$ and the second separator is mapped to to arbitrary vertex in T; the last ${V_T}^2$ columns list the states that vertices in $Cut[i]$ are mapped to arbitrary vertices in T (An example is shown in Fig. 6.5).

Filling DP Table.

The table DT is filled up-bottom. Let us pick the subtree in TD and rooted by $i \in [1..|I|]$ to find its local optimal mapping result. The tree node has $Cut[i] = \{u, v\}$ and $Own[i] = \{w\}$. Its child tree node j has the following possibilities: 1) $Cut[j] = \{w\}$; 2) $Cut[j] = \{u, w\}$; 3) $Cut[j] = \{u, w\}$ and strictly there exists an edge in P with endpoints u and w; 4) $Cut[j] = \{v, w\}$; 5) $Cut[j] = \{v, w\}$ and strictly there exists an edge in P with endpoints v and w.

$C[i, u, f(u), v, f(v)]$ is defined as the value of an optimal solution. When filling the row r_i except the first column, the mapping conditions of separators $Cut[i]$ are fixed. We search all mapping possibilities $V_T \cup \{\mathbf{b}, \mathbf{d}\}$ of owners $Own[i]$. $B[i, u, f(u), v, f(v), w, f(w)]$ is defined as the value of the possible alignment with the fixed mapping of vertices in the tree node. By combining the mapping state of $Cut[i]$ with the one of $Own[i]$, we can decide the best alignment of the child tree nodes of i as well as which entries in previous rows can make contribution to the optimal alignment. Among the alignment possibilities, we choose the minimum cost one. It is the hierarchical decomposition of the underlying pattern graph that characterizes the structure of the optimal solution. The recursive function is as follows:

$$C[i, u, f(u) = \mathbf{d}/u' \in V_T, v, f(v) = \mathbf{d}/v' \in V_T] = \min \begin{cases} B[i, u, f(u), v, f(v), w, \mathbf{d}] \\ B[i, u, f(u), v, f(v), w, \mathbf{b}] \\ min_{w' \in V_T} B[i, u, f(u), v, f(v), w, w'] \end{cases}$$

We know that $dist_{\mathbf{d},w}(v', w') = 0$, $C[j, w, w'] = C[j, w, w', v, d]$, and $C[i, u, \mathbf{d}, v, \mathbf{d}] = down[i]$.

When one of separators is mapped to $\mathbf{d}$, i.e. $f(v) = \mathbf{d}$, we have

$$B[i,u,u' \in V_T, v, f(v) = \mathbf{d}/v' \in V_T, w, \mathbf{d}] \;=\; \Delta(w,\mathbf{d}) + \sum_{Cut[j]=\{w\}} down[j] + \sum_{Cut[j]=\{u,w\}} C[j,u,u',w,\mathbf{d}] + \sum_{Cut[j]=\{v,w\}} C[j,v,f(v),w,\mathbf{d}]$$

When no separators is mapped to $\mathbf{d}$, we have

$$\begin{aligned} B[i,u,u' \in V_T, v, f(v) = \mathbf{d}/v' \in V_T, w, w' \in V_T] \;=\; & \Delta(w,w') \\ & + dist_{u,w}(u',w') + dist_{v,w}(v',w') \\ & + \sum_{Cut[j]=\{w\}} C[j,w,w'] \\ & + \sum_{Cut[j]=\{u,w\}} C[j,u,u',w,w'] \\ & - \sum_{Cut[j]=\{u,w\} \& \exists e(u,w) \& dist(u',w')=\infty} C[j,u,u',w,w'] \\ & + \sum_{Cut[j]=\{v,w\}} C[j,v,f(v),w,w'] \\ & - \sum_{Cut[j]=\{v,w\} \& \exists e(v,w) \& dist(v',w')=\infty} C[j,v,v',w,w'] \end{aligned}$$

The local optimal alignment score $opt[i]$ is calculated as follows:

$$opt[i] = up[i] + \min_{f(u),f(v)} (\Delta(u,f(u)) + \Delta(v,f(v)) + C[i,u,f(u),v,f(v)])$$

The global optimal mapping score is calculated by $OPT = min_{i \in [1..|I|]} opt[i]$.

Runtime Analysis.

As we mentioned earlier, the runtime for constructing the transitive closure $T' = (V_T, E_{T'})$ is $O(|V_T||E_T|)$. The runtime to construct the decomposition tree of pattern is $O(V_P^2)$.

The runtime to fill a cell $DT[i,j]$ is proportional to

$$t_{ij} = (|V_T| + 2) deg_P(v_i) deg_{T'}(u_j)$$

where $|V_T| + 2$ is the number of mapping possibilities of w such as w is mapped to **d**, **b**, or some vertex in T. $deg_P(v_i)$ and $deg_{T'}(u_j)$ are degrees of v_i and u_j in graphs P and T', respectively. Indeed, the number of children of v_i is $deg_T(v_i) - 1$ and for each child v_{i_l} of v_i there are at most $deg_{G'}(u_j)$ feasible positions in G' since $f(v_i)$ and $f(v_{i_l})$ should be adjacent. The number of entries we need to fill is

$$O(|V_P|(3 + tw * |V_T| + |V_T|^{tw}))$$

where tw is equal to the tree width 2 of the specific series-parallel graph and the first item is the size of tree nodes in the worst case.

The runtime to fill the entire table DT is proportional to

$$\sum_{j=1}^{|V_P|} \sum_{i=1}^{3+tw*|V_T|+|V_T|^{tw}} t_{ij} = \sum_{j=1}^{|V_P|} deg_{T'}(u_j) \sum_{i=1}^{3+tw*|V_T|+|V_T|^{tw}} deg_P(v_i) = O(|E_{T'}||E_P||V_T|^{tw})$$

Thus the total runtime is $O(|V_T||E_T| + |E_{T'}||E_P||V_T|^{tw})$. Even though T is sparse, $|E_{T'}|$ may be as large as $O(|V_T|^2)$, i.e., the runtime is $O(|V_T||E_T| + |E_P||V_T|^{tw+2})$.

6.2.4 Results

We have conducted the following experiments to test our alignment algorithms. First we pick up the pathway called de novo biosynthesis of pyrimidine deoxyribonucleotides from the Bacillus subtilis species.

The first experiment is to do self alignment(See Fig. 6.6). In all these alignment result figures, labels with a light gray background represent the pattern vertices and those with a white background represent text vertices; the node with upper part and lower part represents a vertex-to-vertex mapping. The upper part represents the query enzyme and the lower part represents the text enzyme.

The second experiment is to test the text vertex deletions and the pattern vertex deletions (see Fig. 6.7 and 6.8 respectively).

The third experiment is to test the resolution of the ambiguity of databases. We conduct the alignment on the original graphs (See Fig. 6.9). We randomly change a vertex label with the dashed label so that a label is fuzzed. By aligning the fuzzed graph with the well known one, we find the potential candidate (see Fig. 6.10).

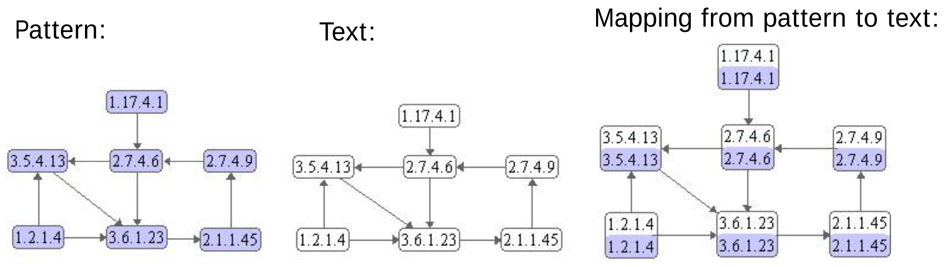


Figure 6.6. Mapping sample with self alignment.

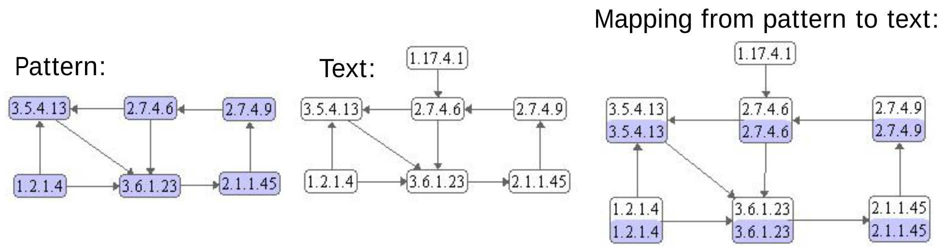


Figure 6.7. Mapping sample 1 with text vertex deletion.

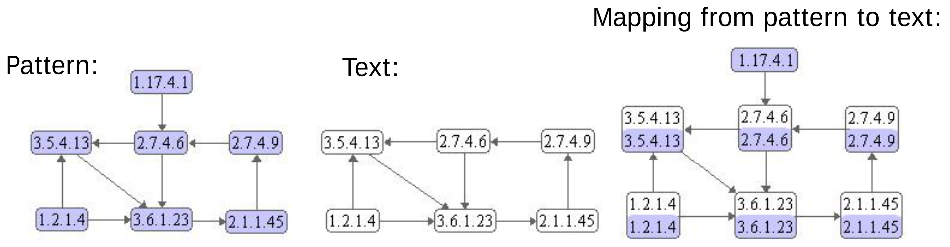


Figure 6.8. Mapping sample 1 with pattern vertex deletion.

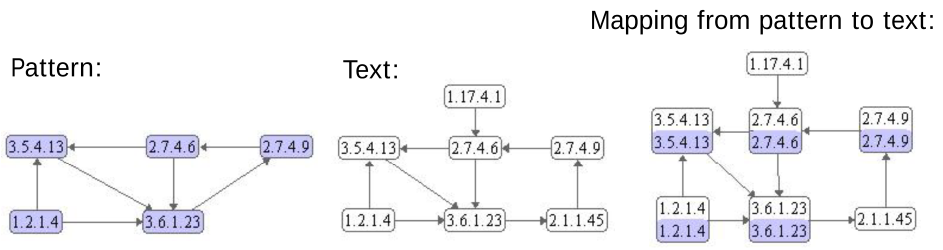


Figure 6.9. Mapping sample 2 with text vertex deletion.

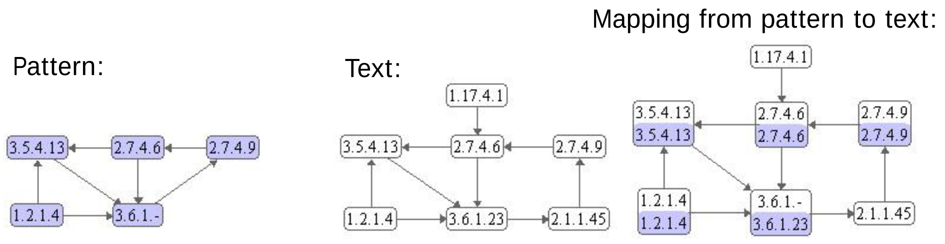


Figure 6.10. Mapping sample 2 with ambiguous label.

CHAPTER 7

METNETALIGN: A WEB SERVICE TOOL

Metabolism is a vital cellular process whose understanding is critical to human disease studies and drug discovery. The accumulation of high-throughput genomic, proteomic and metabolical data allows for increasingly accurate modeling and reconstruction of metabolic networks. Comparison among the reconstructed networks can catch model inconsistencies and infer missing elements. With the growth of identified metabolic networks, computational tools are necessary for the comparison. Network alignment is convenient for comparing and exploring metabolic networks – it can be used for predicting unknown or alternative pathways and pathway holes as well as resolving ambiguities and finding inconsistencies in existing pathway descriptions.

A metabolic pathway/network can be represented as a directed graph which vertices are enzymes. Each of its edges connects enzymes catalyzing consecutive reactions.

We have provided the MetNetAligner web service tool available at:
http://alla.cs.gsu.edu:8080/MinePW/pages/gmapping/GMMain.html .
MetNetAligner relies on the fast dynamic programming based algorithms for metabolic network alignment. The algorithms find an optimal alignment between arbitrary pattern and text graphs allowing enzyme deletion and insertion as well as matching similar enzymes. They applied feedback vertex set and tree decomposition techniques respectively. The feedback vertex set techniques based algorithm is efficient for arbitrary text graphs and pattern graphs with a restricted cyclic structure; the tree decomposition techniques based algorithm is efficient for arbitrary text graphs and pattern graphs with a bounded treewidth. Note that

mimicking evolutionary machinery of gene duplication ([48]), similarly to [61] our algorithm allows to map different pattern enzymes into the same text enzyme.

MetNetAligner provides simple and intuitive web-interfaces and several services such as pathway retrieval, visualization and upload services. Below, we briefly describe our alignment algorithm, present the web service tool, and give usage examples. The algorithm validation and discussion are in supplementary materials.

First I will introduce the architecture, features, and implementation of our web service design; then I will give the manual for the main features;

7.1 Architecture, features, and implementation

Graph comparison is the fundamental computational approach because 1) graph can be used to represent the process of gene or protein interaction or biological network and 2)graph comparison can be used to transfer knowledge from a well-understood network to an unknown one. By using the comparison tools, biologists can save time and energy in the analysis and prediction. Network alignments are useful for comparing and exploring biological pathways. It can be used for predicting unknown and partially known pathways identifying conserved pathways, indicating potential pathway holes and alignment gaps in existing pathways.

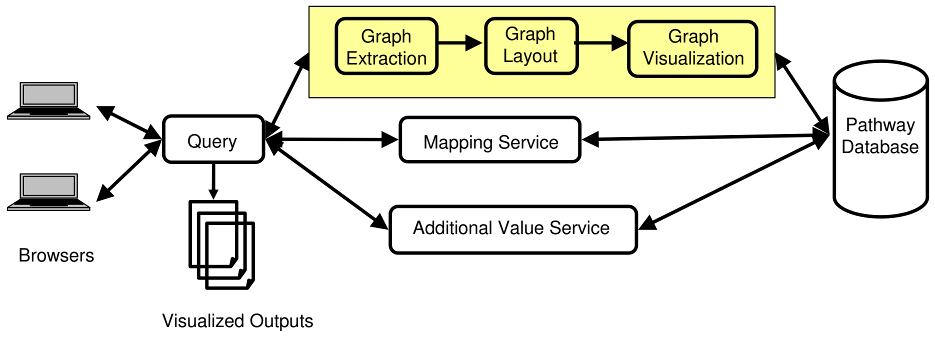


Figure 7.1. Soft architecture.

Network Comparison is extensively used for clustering, mining, and predicting kinds of networks such as metabolic pathways, are essential for graph database search. A series of algorithms have been studied; and kinds of softwares have been developed for the graph comparison based on different research focus and views on specific applications.

A service-oriented graph search and mining tool has been required which will provide great convenience for biologists and facilitate the speedup of their research such as identifying and filling pathway holes. We show our preliminarily designed architecture as shown in figure 7.1. It will consist of 3 ties with four services such as query service, graph visualization service, mapping service and additional value service, during which visualization view of the mapping results with appropriate classification by the graph mining rules will be obtained. We have employed Tomcat/Apache as our application server. Pathway database is the container of plain files. After deploying the application server, we have worked on graph layout, then mapping service and finally additional value service.

7.1.1 Data Source

The MetNetAligner web service is applicable to pathways from Bio-Cyc database ([2]). The metabolic networks of five organisms (E. coli, S. cerevisiae, B. subtilis, Halobacterium sp. NRC-1, and T. thermophilus) are readily available for aligning. The files with metabolic pathways of other organisms can be obtained from Bio-Cyc and extracted using our supporting tools.

7.1.2 Implementation and features

MetNetAligner employs Tomcat /Apache as the application server and uses MySQL as the data storage server. Our tool is portable to several operating systems and compatible to database servers. A web browser with Java SE 6 installed is required for graph visualization.

The work with MetNetAligner is divided into three phases: first, a set of parameters must be specified that describes the alignment mode, parameters and scoring schemes. Second, a user chooses organisms and pathways for pattern and text from the existing database.

Enzyme-enzyme model of pathway and alignment results will be graphically displayed (See Fig. 7.3). The graph layout is based on a force-directed method ([7]). A user can customize the visualization by dragging vertices with the mouse. Labels with a light gray background represent the pattern vertices and those with a white background represent text vertices. Third, for the batch modes, the best K alignment results are sorted by P-value or alignment score. The options, parameters and modes are specified as follows.

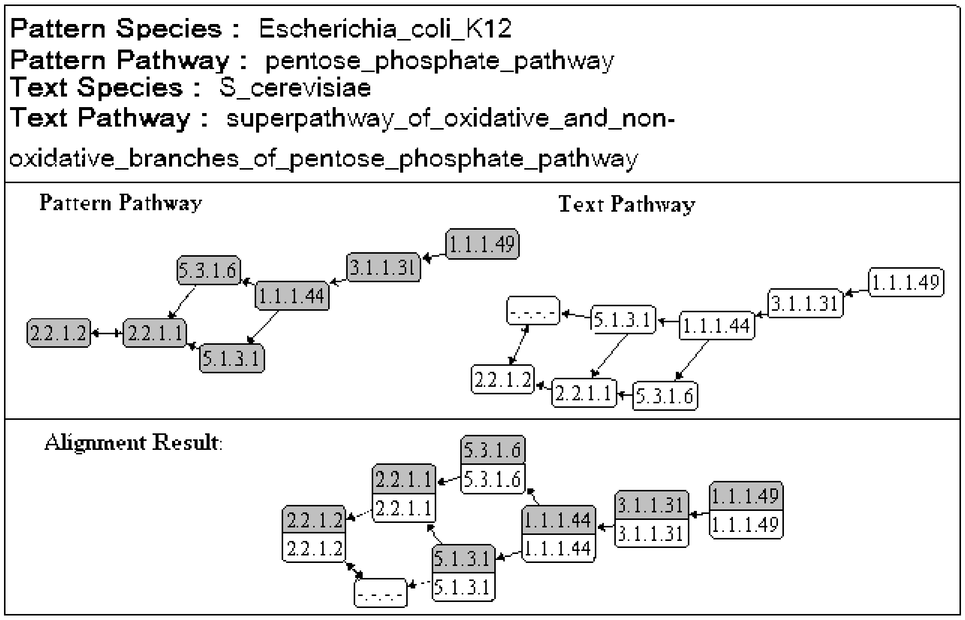

Pattern Species : Escherichia_coli_K12
Pattern Pathway : pentose_phosphate_pathway
Text Species : S_cerevisiae
Text Pathway : superpathway_of_oxidative_and_non-oxidative_branches_of_pentose_phosphate_pathway

Figure 7.2. A sample alignment. The upper part is the name of organisms and pathways of pattern and text; the middle part is a sample pair for the alignment; the bottom part is the alignment result.

Alignment options.

- Allow/forbid enzyme deletion and insertion;

- EC-notation based dissimilarity score from [19]. EC-common-upper-class based dissimilarity score from [52].

- Models of random graph for P-value computation: edge reshuffling and vertex label reshuffling;

Alignment parameters.

- Deletion penalties for pattern or text vertices;

- Single enzyme insertion penalty λ - Balance coefficiency λ. Let A denote the penalties for calculating the influence of enzyme dissimilarity to alignment; let B denote the penalties for calculating the influence of topology dissimilarity; correspondingly, λ represents the balance coefficiency between A and B;

Supported batch modes for pathway alignments.

- *Alignment of two pathways.* The alignment result is visualized (see an example on Fig. 7.3). The visualization can be customized in a drag and drop manner and uploaded to the web tool.

- *Alignments from a single pattern pathway to all pathways in the text organism.* The best K alignments (K is user-specified) are displayed sorted according to P-value or alignment score.

- *Alignments from every pathway in the pattern organism to every pathway in the text organism.* The output gives K most similar pairs of pathways from the two organisms.

The best K alignments (K is user-specified) are displayed sorted according to P-value or alignment score.

7.2 Manual

In this section, we will describe the possible interactions with users. The interactions are related with all input requirements such as how to input user's pathway, how to input pathway from BioCyc, and how to do network alignment in different modes.

The MetNetAligner has been tested on the FireFox 2.0.0.6 version, Internet Explorer 7, and Safari 4 public beta in Windows XP.

7.2.1 How to input user's pathway?

Use any text editor to edit and save your own pathway with the graph file format (see the format here). The pathway file contains only a set of EC notations representing enzymes and a set of pairs representing the relationship of enzymes.

After users define their own pathway, please upload the pathway by entering here.

7.2.2 How to input pathway from BioCyc?

First, request the license (here), download BioCyc data, unzip it;

Secondly, download our supporting tool (here), unzip the downloaded supporting tool, read its readme.txt, run the pahway extraction tool to extract the BioCyc data;

Thirdly, edit users' own pathway based on the resulted pathway;

Fourthly, after users define their own pathway, please upload the pathway by entering here.

7.2.3 How to compare two pathways?

First, click on "Configuration" button in the left frame of the main web page to set up all parameters of pathway alignments. If users do not change any parameter, the pathway alignment will be done based on the default settings. Once users change parameters and confirm their change by clicking "Next", they can use their settings in all available sessions.

Secondly, click on "Mapping" button in the left frame of the main web page to do pathway alignments. Choose pattern and text organism (If users want to find the pathways they uploaded, please choose "$User_Defined$"; Click "Next" button; Then choose pattern and text pathway; Click "Next" button and they will see the patter and text pathway; if they are not satisfied with the layout of the pathways, click on "Visualization" and they will see a new window with the drag-and-drop visualization tool; they could change the layout by dragging or dropping the mouse and save the layout by clicking "submit" button;

Thirdly, click "Align" button; it will popup a new window with the alignment result; if users are not satisfied with the layout of the pathways, click on "Visualization" and they will see a new window with the drag-and-drop visualization tool; they could change the layout by dragging or dropping the mouse and save the layout by clicking "submit" button.

7.2.4 How to compare a pathway with all pathways in an organism ?

First, click on "Configuration" button in the left frame of the main web page to set up all parameters of pathway alignments. If users do not change any parameter, the pathway alignment will be done based on the default settings. Once they change parameters and confirm their change by clicking "Next", they can use their settings in all available sessions.

Secondly, click on "Mapping" button in the left frame of the main web page to do pathway alignments. Choose pattern and text organism (If users want to find the pathways they uploaded, please choose $User_Defined$) ; Click "Next" button; Then choose a pattern pathway and choose "Entire Network" from the text pathway selection list box; Click "Next" button; they will a table which shows the list of mapping results; they could choose the number of best mappings for showing pathways pairs and they could also choose the sort rule; After they choose the show modes, click on "Submit" and they can see the results;

Thirdly, users could click on "View" and see a popup new window which showing pathways; they could click on "Align" button on the new window and see the alignment result. (See details in the subsection 7.2.3)

7.2.5 How to compare all pathways between two organisms?

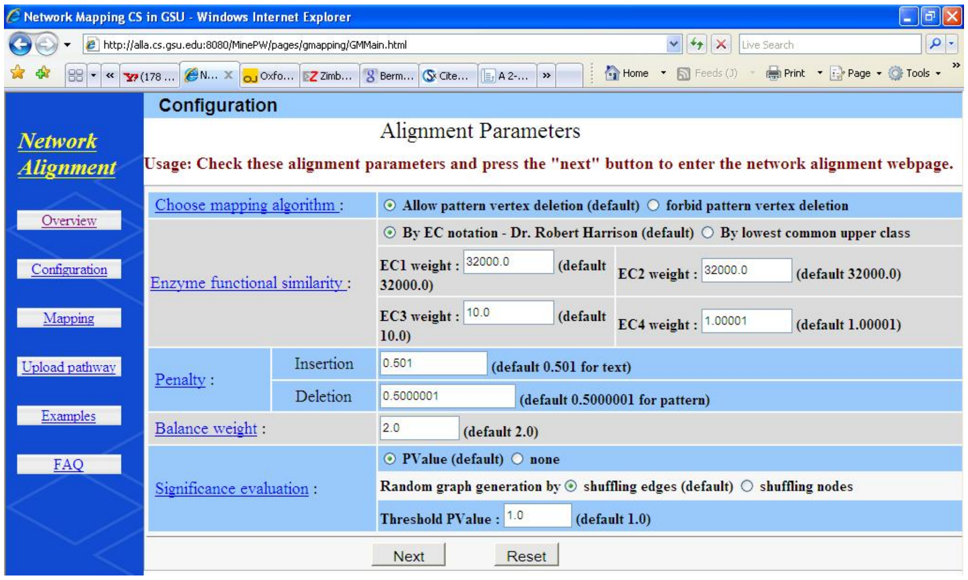
First, click on "Configuration" button in the left frame of the main web page to set up all parameters of pathway alignments. If users do not change any parameter, the pathway alignment will be done based on the default settings. Once they change parameters and confirm their change by clicking "Next", they can use their settings in all available sessions.

Secondly, click on "Mapping" button in the left frame of the main web page to do pathway alignments. Choose pattern and text organism (If users want to find the pathways they uploaded, please choose "$User_Defined$") ; Click "Next" button; Choose "Entire Network" from the pattern and text pathway selection list box; Click "Next" button (It will take time due to the hardware limitation); they will see a table which shows the list of mapping results; they could choose the number of best mappings for showing pathways pairs and they could also choose the sort rule; After they choose the show modes, click on "Submit" and they can see the results;

Thirdly, users could click on "View" and see a popup new window which showing pathways; they could click on "Align" button on the new window and see the alignment result. (See details in the subsection 7.2.3)

7.3 Validation and discussion

The sample input consists of metabolic pathways for five different organisms (*E. coli*, the yeast *S. cerevisiae*, the eubacterium *B. subtilis*, the archeabacterium *T. thermophilus* and the halobacterium *H.NRC-1*), in which exist partially identified or unidentified enzymes. We align every pattern and text pathway using default parameters (see Fig. 7.3) and identify conserved patterns (see Table 7.2) and suggest possible resolutions of the ambiguity of some enzymes (which EC-notation has up to 4 unidentified EC categorization, for example, 1.2.4.− or −. − . − .− (see Table 7.1).


Network Mapping CS in GSU - Windows Internet Explorer
http://alla.cs.gsu.edu:8080/MinePW/pages/gmapping/GMMain.html
Network Alignment
Overview
Configuration
Mapping
Upload pathway
Examples
FAQ
Next
Reset

Configuration

Alignment Parameters

Usage: Check these alignment parameters and press the "next" button to enter the network alignment webpage.

Choose mapping algorithm :		[✓ Allow pattern vertex deletion (default)] forbid pattern vertex deletion
Enzyme functional similarity :		[✓ By EC notation - Dr. Robert Harrison (default)] By lowest common upper class
		EC1 weight : 32000.0 (default 32000.0) EC2 weight : 32000.0 (default 32000.0)
		EC3 weight : 10.0 (default 10.0) EC4 weight : 1.00001 (default 1.00001)
Penalty :	Insertion	0.501 (default 0.501 for text)
	Deletion	0.5000001 (default 0.5000001 for pattern)
Balance weight :		2.0 (default 2.0)
Significance evaluation :		[✓ PValue (default)] none
		Random graph generation by [✓ shuffling edges (default)] shuffling nodes
		Threshold PValue : 1.0 (default 1.0)

Figure 7.3. Configuration web page with default parameters.

Table 7.2 identifies conserved patterns and reports the distribution of indels and mismatches for alignments from *E. coli* to *S. cerevisiae*. For instance, there are 275 pathway pairs for which there is a statistically significant alignment ($p \leq 0.05$) and the number range of found indels and feasible mismatches is 1 through 5. Data curating is the complete copying of pathways; therefore, conserved pathways which are not completely matched cannot be the result of data curating. Thus our method finds both curated and novel network motifs.

Table 7.1 summarizes the following experiments. We randomly erase EC-notations of 10% or 20% vertices for each pathway of a single species, predict their values by alignments, and report their correctly filled ratio classified by the hierarchical sub classes of EC-notations. The experimental study indicates that the alignment with the above discussed deletions is more helpful for the effective prediction on erased EC-notations than the one without pattern vertex deletion.

Table 7.3 illustrates advantage of the network alignemnt (HH) over homomorphisms H (network alignment without vertex deletion). For both characteristics – number of mismatches and number of gaps – the best homo-homeo morphism significantly outperform the best homomorphisms.

Table 7.1. Accuracy ratio of filling erased EC-notations under the aligning with/without pattern vertex deletion from *T. thermophilus* to *E. coli.* The result is classified by the hierarchical sub classes of EC-notations. Here, deletion is specified as pattern vertex deletion. Erasing ratio is 20%.

Alignment	Correctly filled ratio (%)			
	1^{st}	2^{nd}	3^{rd}	4^{th}
W/O deletion	61.9	7.1	7.1	7.1
With deletion	96.3	68.8	51.8	30.5

Table 7.2. Identifying conserved patterns and reporting the distribution of the number of indels and feasible mismatches when aligning *E. coli* to *S. cerevisiae.* Each value is the number of conserved pathway pairs in the range of mismatches and indels and bound on p-value.

P-value	total	Number range of difference (indels + feasible mismatches)					
		0	1-5	6-10	11-15	16-20	≥ 21
≤ 0.01	1083	258	274	117	17	13	372
≤ 0.05	1087	258	275	117	17	13	375
≤ 1	44625	258	397	186	29	51	786

Table 7.3. Alignment of tree pathways from different species with optimal homomorphisms (H) and optimal network alignment (HH). The average number of mismatches and gaps is reported on common statistically significant matched pathways.

	E. coli− >T. thermophilus		E. coli− >B. subtilis		E. coli− >H. NRC-1		E. coli− >S. cerevisiae	
	Mismatches	Gaps	Mismatches	Gaps	Mismatches	Gaps	Mismatches	Gaps
HH	0.58	0.04	0.23	0.03	1.60	0.10	0.22	0.04
H	0.76	0.07	0.38	0.06	2.31	0.12	0.22	0.05

CHAPTER 8
IDENTIFICATION AND FILLING OF PATHWAY HOLES

As more genomes and proteomes are characterized, comparison between them allows us to better understand the evolutionary history of those organisms. Network mapping can be used for comparing and exploring biological pathways, predicting unknown and partially known pathways, indicating conserved pathways across examined species, identifying potential pathway holes or alternative pathways and inconsistencies in existing pathway descriptions. Missing protein annotations cause holes in current metabolic pathway descriptions. The network-mapping tool integrated with protein database search can be used for filling pathway holes. A metabolic pathway under consideration (pattern) is mapped into a known metabolic pathway (text), to find homologous pathways. Pattern pathways with incompletely or mistakenly classified enzymes can be updated based on the annotations of the corresponding text enzyme. Text enzymes that do not show up in the pattern may be a hole in the pattern pathway or an indication of alternative pattern pathways. Based on the results of network alignments, we can identify the pathway holes and indicate the potential candidates of filling the holes. This is becoming a complementary approach to predict protein function.

In this chapter, I will first introduce the definition and types of pathway holes. Then I present how the network alignment can help find the pathway holes and suggest the corresponding candidates of filling the holes. Finally I describe other existing approaches to predict protein function.

8.1 Definition of pathway holes

We distinguish two types of pathway holes.

1. **Visible pathway holes:** an enzyme with partially or completely unknown EC notation (e.g., 1.2.4.- or -.-.-.-) in the currently available pathway description. This type of holes is caused by ambiguity in identifying a gene and its product in an organism.

2. **Hidden pathway holes:** an enzyme that is completely missed from the currently available pathway description. This type of holes occurs when the gene encoding an enzyme is not identified in an organism's genome.

8.2 Identify and fill pathway holes

Mapping of an incomplete metabolic network of a pattern organism into a better known metabolic network can identify possible hidden pathway holes in the pattern as well as suggest possible candidates for filling visible and identified hidden pathway holes.

The candidates for filling pathway holes may have been previously identified in the pattern organism. Then the pathway description with visible holes should be simply updated. In case of hidden holes, the adjoining enzymes may have been incorrectly annotated. Otherwise, if candidates for filling pathway holes have not been identified in the pattern organism, then the adjoining enzymes can be searched for prosites to match the corresponding text enzyme.

The proposed framework for filling pathway holes is based on amino acid sequence homology and, therefore, should be superior to existing frameworks based on DNA homology [36, 42], since amino acids may be coded by multiple codons. Below we apply our framework to two examples of such holes.

Let us analyze an example of how one can fill a visible pathway hole. The alignment of glutamate degradation VII pathway in *B. subtilis* with glutamate degradation VII pathway in *T. thermophilus* is shown in Figure 5.1. The pattern contains two visible pathway holes (the corresponding enzymes are shaded). The mapping results indicate that similar corresponding enzymes 2.3.1.61 and 1.2.4.2 with similar functions can be found in *T.thermophilus*. Our tool queries the Swiss-Prot and TrEMBL databases to see if enzyme 2.3.1.61 and 1.2.4.2 have been reported for *B. subtilis* We found that these two enzymes have been reported in Swiss-Prot database for *B. subtilis* as P16263 and P23129 respectively. Therefore we recommend filling these pathway holes with enzymes 2.3.1.61 and 1.2.4.2.

Now we will proceed with an example of a hidden pathway hole. Mapping of formaldehyde oxidation V pathway in *B. subtilis* to formy1THF biosynthesis pathway in *E. coli* is shown in Figure 5.3. In this case the enzyme 3.5.1.10 is present between 3.5.4.9 and 6.3.4.3 in *E. coli*, but absent in the pathway description for *B. subtilis*. The Swiss-Prot database search shows that this enzyme is completely missing from *B. subtilis* and therefore this hole does not allow an easy fix. Still it is possible that this enzyme has not yet been included in the database but has been already identified either in the literature (this can be detected through keyword search) or in closely related organisms. The Swiss-Prot database search shows that 3.5.1.10 has been reported for *B.clausii* which is very close to *B. subtilis*. Therefore we recommend filling this pathway hole with enzymes 3.5.1.10. If such search would not return hits in close relatives, then we would investigate if the function of this enzyme has been taken up by one of the adjoining enzymes (in this case 3.5.4.9 or 6.3.4.3), or there is an alternative pathway existing for this function.

8.3 Practical application

Dip used our alignment tools in her thesis which is under the direction of Dr. Zelikovsky. In her work, she read the the ambiguity and holes report from our tool and designed a automatic flow to validate the potential candidate. In the flow, the first step is to check if there is such an enzyme in organism. If such an enzyme exists, it agrees that the prediction of our tool is correct and some inconsistency or mistakes occur in databases; If there is no such enzyme, it will align neighbor enzymes. It is because functions may be taken over; If it find nothing, it will search for the closest protein in the same group to text enzyme. If identity is too high, i.e. more than 50, it identifies the potential candidates; If nothing is found, it will align every enzyme in paths with whole genome. In her thesis, she listed a series of potential candidates which resulted from our tool usage. The identified candidates resolved ambiguities and can be used to fill the holes.

BIBLIOGRAPHY

[1] http://people.math.gatech.edu/ thomas/teach/8863/serpar.pdf.

[2] http://www.biocyc.org/.

[3] http://www.genome.jp/kegg/pathway.html.

[4] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM 42 (1995), 844-856.*

[5] R. Ambauen, S. Fischer, and H. Bunke. Graph edit distance with node splitting and merging and its application to diatom identification. *IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition, LNCS, Springer Verlag, 95- 106*, 2003.

[6] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math. 12(3): 289-297*, 1999.

[7] J. Barnes and P. Hut. A hierarchical o(n log n) force calculation algorithm. *Nature, v.324*, 1986.

[8] Hans Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *Proceedings 22nd International Colloquium on Automata, Languages and Programming*, 1995.

[9] Hans Bodlaender and Arie Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal Advance Access published July 19, 2007*, 2007.

[10] Hans L. Bodlaender and Babette de Fluiter. Parallel algorithms for series parallel graphs. *Proceedings 22nd International Colloquium on Automata, Languages and Programming*, 1995.

[11] Karsten Michael Borgwardt. Graph kernels. *PhD thesis in Computer Science, Ludwig-Maximilians-University Munich*, 2007.

[12] Sharon Bruckner, Falk Hffner, Richard M. Karp, Ron Shamir, and Roded Sharan. Topology-free querying of protein interaction networks. *In Proceedings of the 13th Annual International Conference on Research in Computational Molecular Biology (RECOMB '09)*, 2009.

[13] H. Bunke. Error-tolerant graph matching: A formal framework and algorithms. *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition. Lecture Notes in Computer Science. Springer, Berlin*, 1998.

[14] Horst Bunke. Graph matching: Theoretical foundations, algorithms, and applications.

[15] M. Chen and R. Hofestaedt. Pathaligner: metabolic pathway retrieval and alignment.

[16] Ming Chen and Ralf Hofest. An algorithm for linear metabolic pathway alignment. *In silico biology (In silico biol.) ISSN, 1386-6338: 111-128*, 2005.

[17] Q. Cheng, P. Berman, R. Harrison, and A. Zelikovsky. Fast alignments of metabolic networks. *BIBM*, 2008.

[18] Q. Cheng, R. Harrison, and A. Zelikovsky. Metnetaligner: a web service tool for metabolic network alignments. *Bioinformatics. Advanced Access published May 4*, 2009.

[19] Qiong Cheng and Alexander Zelikovsky. Optimal mapping of multi source trees into dag in biological network. *ISBRA'07Poster*, May 2007.

[20] Fan Chung, Linyuan Lu, Gregory Dewey, and David Galas. Duplication models for biological networks. *J. Comp. Biol., 10, 677687*, 2003.

[21] L. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(10):1367.1372*, 2004.

[22] Banu Dost, Tomer Shlomi, Nitin Gupta, Vineet Bafna, and Roded Sharan. Qnet: A tool for querying biological networks. *RECOMB 2007 and JCB 2008.*

[23] A. Ferro, R. Giugno, A. Pulvirenti, and et al. Multi-feature graph database searching. *IEEE Transaction Pattern Analysis and Machine Intelligence*, 2005.

[24] Babette Fluiter and Hans Bodlaender. Parallel algorithms for treewidth two. *Tech. report UU-CS-1997-23, Univ. Utrecht, Dept. of Computer Science, Jul 1997*, 1997.

[25] P. Foggia, C. Sansone, and M. Vento. A performance comparison of five algorithms for graph isomorphism. *Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, pages 188-199*, 2001.

[26] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.

[27] R. Giugno and D. Shasha. Graphgrep: A fast and universal method for querying graphs. *ICPR*, 2002.

[28] Michelle L Green and Peter D Karp. A bayesian method for identifying missing enzymes in predicted metabolic pathway databases. *BMC Bioinformatics*, Sep. 2004.

[29] Jing-Dong Jackie Han. Understanding biological functions through molecular networks. *Cell Research 18:224-237.*, 2008.

[30] F. Hffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *The Computer Journal.*

[31] M. et al. Kanehisa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Res., 34, D354D357*, 2006.

[32] I. M. Keeler, V. J. Collard, C. S. Gama, J. Ingrafts, S. Palely, I. T. Paulson, M. Peralta-Gil, and P. D. Karp. Ecocyc: a comprehensive database resource for escherichia coli. *Nucleic Acids Research, 33(1):D334-337*, 2006.

[33] Brian P. Kelly, Roded Sharan, Richard M. Karp, Taylor Sittler, David E. Root, and Brent R. Stockwell. Pathblast: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, Vol.32 : W83-W88, 2004.

[34] Brian P. Kelly, Roded Sharan, Richard M. Karp, Taylor Sittler, David E. Root, and Brent R. Stockwell. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *PNAS*, 11394-11399, Sep. 30 2003.

[35] N. Ketkar, L. Holder, D. Cook, R. Shah, and et al. Subdue: Compression-based frequent pattern discovery in graph data. *ACM KDD*, 2005.

[36] Vitkup D. et al Kharchenko P., Chen L. Freund Y. Identifying metabolic enzymes with multiple types of association evidence. *BMC Bioinformatics*, 2006.

[37] M. Koyuturk, A. Grama, and W. Szpankowski. Pairwise local alignment of protein interaction networks guided by model evoluation. *Journal of Computational Biology, 13, 182-199*, 2006.

[38] C. J. Krieger, P. Zhang, L. A. Mueller, A. Wang, S. Paley, M. Arnaud, J. Pick, S.Y. Rheme, and P.D. Karp. Metacyc: a multiorganism database of metabolic pathways and enzymes. *Nucleic Acids Research, 32(1):D438-42*, 2006.

[39] Zhenping Li, Yong Wang, Shihua Zhang, Xiang-Sun Zhang, and Luonan Chen. Alignment of protein interaction networks by integer quadratic programming. *EMBS '06. 28th Annual International Conference of the IEEE*, 5527-5530, Aug. 2006.

[40] B. T. Messmer. Efficient graph matching algorithm for preprocessing model graphs.

[41] A. Mithani, G. Preston, and J. Hein. Rahnuma: Hypergraph based tool for metabolic pathway prediction and network comparison. *Bioinformatics*, 2009.

[42] Kharchenko P., Vitkup D., and Church G.M. Filling gaps in a metabolic network using expression information. *Bioinformatics, Suppl 1:i178-85*, 2004.

[43] Ron Y Pinter, Oleg Rokhlenko, Esti Yeger-Lotem, and Michal Ziv-Ukelson. Alignment of metabolic pathways. *Bioinformatics.*

[44] R.Y. Pinter, O. Rokhlenko, D. Tsur, and M. Ziv-Ukelson. Approximate labeled subtree homeomorphism. *In Proceedings of 15th Annual Symposium of Combinatorial Pattern Matching.*

[45] Erzsebet Ravasz and Albert-Laszlo Barabasi. Hierarchical organization of modularity in metabolic networks. *Physical Review E67, 026112*, 2003.

[46] N. Robertson and P.D. Seymour. Graph minors. ii. algorithmic aspects of treewidth. *J. of Algorithms 7, 309322*, 1986.

[47] Berry Schoenmakers. A new algorithm for the recognition of series parallel graphs. *CWI Report CS-R9504, January 1995*, 1995.

[48] R. Sharan and T. Ideker. Modeling cellular machinery through biological network comparison.

[49] Roded Sharan, Silpa Suthram, Ryan M. Kelley, Tanja Kuhn, Scott McCuine, Peter Uetz, Taylor Sittler, Richard M. Karp, and Trey Ideker. Conserved patterns of protein interaction in multiple species. *PNAS*, Vol.102 : 1974-1979, 2005.

[50] D. Shasha, J.T-L Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. *PODS, pages 39.52*, 2002.

[51] O. A. Shcherbina. Tree decomposition and discrete optimization problems: A survey. *Cybernetics and Systems Analysis E67, 026112*, 2007.

[52] Y. Tohsato, H. Matsuda, and A. Hashimoto. A multiple alignment algorithm for metabolic pathway analysis using enzyme hierarchy. *Proc. 8th International Conference on Intelligent Systems for Molecular Biology*, 376-383, ISMB 2000.

[53] W.H. Tsai and K.S Fu. Error-correcting isomorphisms of attributed relational graphsfor pattern recognition. *Systems, Man, and Cybernetics, 9:757-768*, 1979.

[54] J. Valdes, R. Tarjan, and E. Lawler. The recognition of series parallel digraphs. *SIAM J. Comput. 11 2 (1982), pp. 298313*, 1982.

[55] Sebastian Wernicke. Combinatorial algorithms to cope with the complexity of biological networks. *Dissertation*, December 2006.

[56] Sebastian Wernicke and Florian Rasche. Simple and fast alignment of metabolic pathways by exploiting local diversity. *Bioinformatics 23(15):1978*, 2007.

[57] Gerhard J. Woeginger. Exact algorithms for np-hard problems: A survey. *Combinatorial Optimization (Edmonds Festschrift), LNCS 2570, pp. 185207, 2003*, 2003.

[58] Y. Y. Tian, R. C. McEachin, C. Santos, D. J. States, and et al. Saga: A subgraph matching tool for biological graphs. *Bioinformatics, 23(2):232-239*, 2007.

[59] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. *ICDM, pages 721-724*, 2002.

[60] X. Yan, P. S. Yuz, and J. Hany. Graph indexing: a frequent structure-based approach. *SIGMOD, pp. 335-46*, 2004.

[61] Qingwu Yang and Sing-Hoi Sze. Path matching and graph matching in biological networks. *Journal of Computational Biology*, Vol. 14, No. 1: 56-67 : 5527-5530, 2007.

[62] M. et al. Yeung. Estimation of the number of extreme pathways for metabolic networks. *BMC Bioinformatics, 8, 363*, 2007.

[63] Anton Yuryev and Sean Ekins. *Pathway Analysis for Drug Discovery: Computational Infrastructure and Applications.* Wiley knowledge for generations, ISBN: 978-0-470-10705-8, 2008.