Algorithms for Labeled Graph Matching
with Applications to Systems Biology

Oleg Rokhlenko
Algorithms for Labeled Graph Matching with Applications to Systems Biology

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Oleg Rokhlenko

Submitted to the Senate of the
Technion — Israel Institute of Technology

Kislev 5768 Haifa December 2007
The Research Thesis Was Done Under the Supervision of Associate Professor Ron Pinter in the Department of Computer Science.

Foremost, I would like to thank my adviser, Ron Pinter, for his continuous guidance, encouragement, comments and support. He was always eager to help, be it a research issue or a personal matter, and his office door was always open to me whenever I needed his advice. I would also like to warmly thank Michal Ziv-Ukelson, with whom I had the pleasure to work, for the generous time and positive attitude she offered me. I sure will miss the brainstorming and creative discussions with Michal over a cup of coffee. Finally, I would regret my doctoral years if I did not meet Ydo Wexler, who became my true friend and shared with me both happy moments as well as difficult times.

This thesis is dedicated to my mother, Valentina Rokhlenko, for her love, never-ending support, and confidence in my abilities.

# Contents

1 **Introduction** 1  
  1.1 The Subtree Homeomorphism Problem 3  
  1.2 Applications and Suitable Similarity Measures 5  
  1.2.1 Biological Similarity Measures 8  
  1.2.2 Computational Similarity Measures 9  
  1.3 Outline 10  

2 **Approximate Labeled Subtree Homeomorphism** 11  
  2.1 Basic Algorithms for ALSH 13  
  2.1.1 Rooted Unordered Trees 13  
  2.1.2 Unrooted Unordered Trees 16  
  2.1.3 Ordered Rooted Trees 19  
  2.2 A More Efficient ALSH Algorithm for Unordered Trees 20  
  2.3 An ALSH Algorithm for Ordered Unrooted Trees 24  
  2.3.1 Cyclically Ordered Unrooted Trees 25  
  2.3.2 Linearly Ordered Unrooted Trees 26  
  2.4 Discussion 27  

3 **Seeded Tree Alignment** 29  
  3.1 Preliminaries and Definitions 30  
  3.2 Tree Alignment based on Seeded Nodes 31  
  3.3 Planar Tanglegram Layout 36  
  3.4 Applications 43  
  3.5 Discussion 47  

4 **Alignment of Metabolic Pathways** 49  
  4.1 Modeling and Computational Approach 50
4.1.1 Extensions to Directed Multi-Source Trees ............................................. 53
4.2 Application to Metabolic Pathway Analysis .............................................. 55
4.3 Results ........................................................................................................ 57
  4.3.1 Inter-species Alignments ........................................................................ 57
  4.3.2 Intra-species Alignments ....................................................................... 59
  4.3.3 MetaPathway Queries ........................................................................... 60
4.4 Discussion ..................................................................................................... 61

5 Functional Similarity Measure between Metabolic Genes ......................... 63
  5.1 Background ................................................................................................. 64
  5.2 Computational Approach ........................................................................... 65
    5.2.1 Modeling Metabolism ........................................................................... 65
    5.2.2 Model-based Similarity Measures for Metabolic Genes ....................... 67
    5.2.3 Topology-based Measure for Metabolic Genes .................................. 72
    5.2.4 Expression-based Measure for Metabolic Genes ................................ 72
    5.2.5 Phylogenetic Profiling Analysis .......................................................... 74
  5.3 Results ......................................................................................................... 74
    5.3.1 Validating the Similarity Measures based on GO ................................. 74
    5.3.2 Validating the Similarity Measures based on Gene Expression Data .... 77
  5.4 Discussion ................................................................................................... 78

6 Conclusions ..................................................................................................... 81

Bibliography ...................................................................................................... 85


APPENDIX B: Studying Regulatory Mechanisms in Yeast using Environmental
             Conditions ................................................................................................. 105
List of Figures

1.1 An example of graph isomorphism. ........................................... 2

1.2 (a) Classical subtree homeomorphism. (b) Approximate Labelled Subtree Homeomorphism. The matching score is explained in Chapter 2. ........................................... 4

1.3 RNA secondary structure and its tree representations. (a) RNA structure, dots representing the nucleic acids; (b) A relatively rough tree representation of this structure where the following labels are used: a node labeled S(tem) represents a sequence of paired bases, a node labeled B a bulge, a node labeled I an inside loop, a node labeled M a multi-branch loop, and a node labeled H a hairpin; (c) A coding of the same structure where each node is a pair of unpaired nucleotides. ........................................... 6

1.4 Lactate biosynthesis pathway. (a) Visual diagram from the MetaCyc database. (b) Enzyme graph representation. Each node represents an enzyme labeled according to its EC classification. (c) Metabolite graph representation. Each node represents a metabolite labeled by its name. ........................................... 7

2.1 (a) Example for subtree homeomorphism. Two subtrees in the text tree that are homeomorphic to the pattern tree are circled by dashed lines. The white nodes in the text tree are degree-2 nodes that are removed by the homeomorphism. (b) Approximate labeled Subtree Homeomorphism. The node-label similarity scores are specified in Table Δ. Note that δ = −1. Two subtrees in the text that are homeomorphic to the pattern are circled by dashed lines. The left homeomorphic subtree has an LSH score of 5, whereas the right one has an LSH score of 12. ........................................... 12

2.2 The work done by Algorithm 1 during the computation of RScores[v, u]. The bipartite graph is constructed in order to compute the optimal weighted assignment between the children of u and those of v. w_{ij} marks the optimal LSH similarity score for aligning the subtrees rooted at y_j ∈ T and x_i ∈ P. ........................................... 14
2.3 (a) Approximate Weighted Episode Matching Calculation. The matching is computed on the dynamic programming graph with score $-\infty$ on the vertical edges, score 0 on the horizontal edges and $\Delta[x_i, y_j]$ scores on the diagonal edges. (b) Linearly Ordered Unrooted Trees - the calculation of the best possible alignment between $X - x_i$ and $Y$. 

2.4 (a) Graph $G'$ with a bipartition of its vertices into sets $X$ and $Y$, and the compressed graph $G^*$ that contains an additional layer $C$. (b) The mapping $M$ of subunits of sub-paths from $G^*$ into subunits of sub-paths in $G'$. 

2.5 Cyclic and linear orders of unrooted trees. 

2.6 The cyclic string comparison problem. 

3.1 An illustration of the dynamic programming (DP) table computed during the seeded matching algorithm. (left) The matched trees with a partitioning induced by seeds. (right) The corresponding DP table, divided into independent rectangles to be computed by an appropriate LCA-preserving mapping algorithm. The colored areas illustrate which parts of the DP need to be computed. The lowest right-most corner of each colored rectangle holds the value for the roots of the corresponding compared subtrees. Each dashed rectangle corresponds to a secondary seed. Within each dashed rectangle, the single-cell components correspond to seeds, where the primary seeds are at the bottom-right and top-left (only for the subtrees framed by two primary seeds) corners, and the secondary seed is located between the two rectangles, each corresponding to one of the subtrees rooted at this secondary seed. 

3.2 The refinements made by the planar tanglegram layout algorithm on the tanglegram of Example 2 (right) follow one of these two paths until a graph is found that has (under the tanglegram layout constraints) no planar layout. 

3.3 The refinements made by the planar tanglegram layout algorithm on the tanglegram of Example 2 (left) follow one of these paths, until a graph is found that has a planar layout and includes all of the seeded nodes. Red arrows indicate clade flippings. 

3.4 Possible connections for $u_1, u_2$ without crossings (and degree at least 1). The exchange of $u_1$ and $u_2$ would create an edge crossing in all cases. 

3.5 The known secondary structures for two RNase P sequences and the corresponding coarse-grain trees. (left) $E. coli$ RNase P, based on [44], shaded its represent conserved loci. (right) $M. barkery$ RNase P obtained from the RNase P database http://www.mbio.ncsu.edu/RNaseP/
3.6 Seeded tree alignment for the E. coli versus M. barkery RNase P secondary structures shown in Fig. 3.5. Dark vertices represent conserved loci, dotted lines represent alignment seeds.

3.7 Seeded phylogenetic unordered tree alignment, in the context of horizontal gene transfer prediction. (left) The tanglegram formed by connecting, via seed edges, the phylogenetic tree based on archeal RNase P structures [46] with another phylogenetic tree based on archeal RNA [50]. The seed edges for the two direct neighbors, according the RNase P RNA tree: Archeaglogi and Methanococci, which are putatively involved in RNase P RNA horizontal transfer [46], were omitted. (right) The planar layout of the tanglegram.

4.1 The work done by the ALSH algorithm during the alignment of the subtree $P^u$ with the subtree $T^v$. The score for entry $(u, v)$ of the above DP table is computed via the corresponding weighted bipartite graph. The node-label similarity scores used in this example are specified in Table $\Delta$ of Figure 4.2.

4.2 Approximate Labeled Subtree Homeomorphism. For each node, the label is written inside the circle and the variable name assigned to the node is written externally. The node-label similarity scores are specified in Table $\Delta$. Note that the deletion score is set to $-1$. A subtree in the text that is homeomorphic to the pattern is circled by the dashed line. The LSH score for this alignment is 7.

4.3 The top-scoring inter species alignments. Each node represents a match: the upper part represents the query enzyme, and the lower part represents the text enzyme. Color shades reflect enzyme homology. [1] The phenilalanine, tyrosine and tryptophan pathway of E.coli vs. S. cerevisiae ($s = -4.28, p < 0.01$). [2] homoserine and methionine biosynthesis of E. coli vs. S. cerevisiae ($s = -13.15, p < 0.01$). . . . .

4.4 The top-scoring intra species alignments. [1] The isoleucine vs. valine biosynthesys pathways of S. cerevisiae ($s = 0, p < 0.01$) alignment. [2] The trehalose anabolism pathways of S. cerevisiae vs. sucrose biosynthesis pathway of S. cerevisiae ($s = -9.58, p < 0.01$). [3] The tyrosine biosynthesis of E.coli vs. the phenilalanine biosynthesis of E. coli ($s = -8.23, p < 0.01$). . . . . . . .

4.5 A meta-pathway query alignment. [1] A meta-query. [2] The alignment of a meta-query with the ureide degradation pathway of S. cerevisiae (left, $s = 0, p < 0.01$) and with the alantoine degradation pathway of E.coli (right, $s = 0, p < 0.01$). . .
5.1 A schematic illustration of two types of similarity measures between metabolic genes. (a) The Environmental response similarity (ERS) measure. Each element in vector $U(V)$ corresponds to the response of gene $U(V)$ to environment $i$. (b) The Genetic response similarity (GRS) measure. Each element in vector $U(V)$ corresponds to the response of flux $i$ to the knock-out of gene $U(V)$.

5.2 Schematic illustration of the proposed flux similarity model. $w$ stands for the optimal flux distributions on the wild-type metabolic network, $v_1$ stands for the optimal flux distribution on the metabolic networks with the first flux knocked-out, and $v_2$ stands for the optimal flux distribution on the metabolic networks with the second flux knocked-out.

5.3 An overlap between the considered GO terms.

5.4 A Venn diagram displaying the consistency of GO terms under the ERS and GRS measures.

5.5 A Venn diagram presenting the consistency of GO terms under ERS and topology-based similarity (TOBS) vs. the (a) expression-based similarity (EXBS) and (b) conservation-based similarity (COBS) measures.

5.6 The Ergosterol biosynthesis pathway. Each node (rectangle) represents an enzyme (except for the last one (ellipse) representing the final product — ergosterol). Each edge represents a metabolite which is produced by one enzyme and consumed by the following one in the pathway. Since ergosterol biosynthesis is carried out through a long, chain-like pathway, the average distance between genes annotated with this term is significantly high, while a mass-balance constraint determines that all genes should either be coherently active or non-active. Thus ERS outperforms the TOBS topological measure.

5.7 Correlation between co-expression levels (EXBS) and model-based (GRS and ERS) or topology based (TOBS) measures. The correlation is obtained by dividing the EXBS axis into uniform intervals and averaging the corresponding values of GRS, ERS and TOBS in each interval. (a) GRS/ERS measures. (b) TOBS measure.

A.1 The input form.
A.2 The file browser for the Text section.
A.3 The progress bar which appears during the computation.
A.4 The results form.
A.5 The alignment form showing both the pattern and the text.
A.6 The details on node-to-node match.
A.7 The alignment form showing only the pattern.
A.8 The high level design of the prototype tool.
Abstract

Labeled graphs are being used extensively in computational biology to represent entities, such as biochemical networks and pathways, RNA secondary structures, and phylogenetic trees. One of the major challenges is to provide techniques for inexact matching (and the resulting alignment) of such graphs in a way that optimizes some objective function. This function usually measures both the similarity between vertices and between edges of two graphs, as well as the resemblance of their structure.

The inexact graph matching problem is NP-hard, and therefore algorithms that reduce the computational complexity of the matching process by imposing topological restrictions on the graphs are called for. In this case inexact graph matching can be regarded as a constrained combinatorial optimization problem. This thesis analyzes the issues and mechanisms that have to be considered for this purpose, regarding topological restrictions, optimization functions, and similarity measures.

Specifically, we study new optimization techniques for the alignment of labeled trees, present a novel approach that utilizes prior knowledge about the required matching to improve a whole class of optimization algorithms, and apply a particular inexact tree matching problem to the alignment of metabolic pathways as an example of the possibilities that this new tree matching approach offers. In addition, we develop a novel optimization approach for devising functional relations between metabolic genes that can be used also as a similarity measure between enzymes that comprise metabolic pathways.

The described techniques have important applications also in areas other than systems biology.
Chapter 1

Introduction

Graphs are a very powerful computational tool that is widely used in scientific applications. Depending on the requirements of the application, there are numerous techniques for the examination and the analysis of such graphs, e.g. finding shortest paths, detecting Hamiltonian cycles, coloring the vertices (or edges) and many more [21, 28, 92]. While a graph can be either unlabeled or labeled, the latter is a much more powerful model. One of the most challenging problems when using labeled graphs is the comparison of graphs with each other. For example, in computational biology, when graphs are used to represent biochemical networks, the recognition of a known component in an unknown network can be performed by trying to find a correspondence between the graph representing the pattern component and the graph representing the target network. Trying to find such correspondence is generally referred to as graph matching.

There are two major classes of graph matching: exact and inexact. An informal description of these two classes is as follows:

I. **Exact Matching.** In order to achieve a good correspondence between graphs, the most common way is to search for an isomorphism. When matching two labeled graphs $G_1$ and $G_2$ by means of *graph isomorphism* we are looking for a bijective mapping between the vertices of $G_1$ and $G_2$ such that both the labels’ identity and the structure of the edges is preserved by the mapping function. When such a mapping function can be found then $G_1$ and $G_2$ are isomorphic. If one of the graphs involved in the matching process is larger than the other, i.e. w.l.o.g. $G_2$ has more vertices than $G_1$, then we are looking for a *subgraph isomorphism* from $G_1$ to $G_2$. That is, we are interested in finding a subgraph $S$ of $G_2$ such that $G_1$ and $S$ are isomorphic.

II. **Inexact Matching.** In many applications the isomorphism requirements are is too strong, and the problem is expressed rather as an *inexact graph matching* problem where a surjective
mapping between the graphs is not required. This type of graph matching problem does not imply searching for the exact way of matching vertices of one graph with vertices of another, but rather finding the *best* matching between them. Thus, most studies of inexact graph matching rely on the optimization of some objective function. This function usually measures both the similarity between vertices and between edges of two graphs, as well as the resemblance between the structure of two graphs. While the topological resemblance is always revealed by a function from one topological space to another (for example, graph homeomorphism), the vertices/edges similarity is an application dependent property. We will discuss this extensively in due course. Examples of optimization techniques include estimation of distribution algorithms (EDAs) [68], tree search [22], and graph editing [12], to name but a few.

One of the main drawbacks of graph matching — even the exact one, which can be considered as the simpler between the two — is its computational complexity. It is still an open question whether the graph isomorphism problem belongs to the complexity class $P$ or whether it is NP-hard [11, 39]. All algorithms that have been developed so far for the general graph isomorphism problem require in the worst case exponential time. For the subgraph isomorphism problem it is well known that it is NP-complete [39] and — as a matter of fact — a naive graph pattern matching algorithm which (i) generates each possible mapping from the $n$ nodes of $G_1$ to $m$ nodes in $G_2$ and (ii) tests whether these mappings are isomorphisms requires in the worst case $O(m^n)$ time. Similarly, the complexity of the inexact subgraph matching problem is equivalent in complexity to the maximum common subgraph isomorphism problem, which is also known to be NP-complete [39]. Consequently, to date no algorithm could be constructed that guarantees to find subgraph isomorphism in polynomial time. Moreover, allowing label similarity rather than identity imposes an additional computational complication.
Therefore, algorithms having time complexity requirements suited for matching large graphs have been the subject of research during the last three decades. One of the most practical measures is to reduce the computational complexity of the matching process by imposing topological restrictions on the graphs. E.g. there are efficient algorithms for finding isomorphism between planar graphs [49], bounded valence graphs [73], and trees [1]. The latter are of special interest due to the fact that many applications impose tree-like topologies on the objects under study. The matching of labeled trees has many important applications in areas such as bioinformatics, semistructured databases, and linguistics. Examples include the comparison among metabolic pathways, the study of alternative evolutionary trees (phylogenies), processing queries against databases and documents represented in e.g. XML, and many fundamental operations in the analysis of natural (and formal) languages. In all these scenarios, both the labels on the nodes as well as the structure of the underlying trees play a major role in determining the similarity between a pattern and the text in which it is to be found. In order to face the complexity issue, the inexact graph matching problem is regarded as a constrained combinatorial optimization problem. This thesis analyzes some of the issues and mechanisms that must be considered for this purpose and presents novel methods for inexact graph matching on trees.

1.1 The Subtree Homeomorphism Problem

In trees, like in graphs, the matching can be either exact or inexact. For exact tree matching, the basic problem (analogous to subgraph isomorphism) is the **subtree isomorphism problem** [77, 78, 107]: Given a pattern tree $P$ and a text tree $T$, find a subtree of $T$ which is isomorphic to $P$ or decide that there is no such tree. The **subtree homeomorphism problem** [18, 123] is a variant of the former problem, where degree 2 nodes can be deleted from the text tree (see Figure 1.2(a)) and replaced by edges connecting their end-points. The **constrained tree inclusion** problem [120] is a variant of labeled subtree homeomorphism where label equality between pairs of aligned nodes in the compared subtrees is required. Note that all the tree matching problems mentioned so far have efficient algorithms.

The **tree inclusion problem** [64] is the problem of locating all the smallest subtrees of $T$ that include $P$, where a tree is included in another tree if it can be obtained from the latter by deleting nodes. Here, deleting a node $v$ from a tree entails deleting all edges incident upon $v$ and inserting edges connecting the parent of $v$ with the children of $v$. **Tree inclusion** is NP-complete [64]. Schlieder and Naumann [100] extended the tree inclusion problem to an **approximate tree embedding problem** in order to retrieve and rank search results using a tree-similarity measure whose semantics are tailored to XML data. The complexity of their algorithm, which is based on dynamic programming and processes the query tree in a bottom up fashion, is exponential.
The inexact tree matching problem is motivated by the aforementioned applications, where a more meaningful match can be found if — in addition to the structural similarity between subtrees — node-to-node resemblance is also taken into account. For example, in bioinformatics, the similarity between metabolic pathways [82, 88, 102] is based both on the resemblance of the elements constituting the pathways (e.g. enzymes) as well as on the likeness of their network structure. A query that searches for a small pathway fragment in a larger tree should take into account the topological similarities (in the form of subtree homeomorphism) as well as the pairwise similarity between enzymes which make up the aligned subtree nodes. Another example is when designing a semantic query language for semistructured databases that are represented as XML documents [100]: here the node-to-node similarity score may reflect content or tag resemblance, and the topological difference allows flexibility in document structure matching [15]. Finally, in natural language processing, trees are often used to represent sentences, where nodes are labeled by words and sentential forms (or constituents); matching that takes into account synonyms and elisions is very useful in order to detect semantically close phrases.

Thus, this thesis addresses the challenge of extending labeled subtree homeomorphism to a new optimization problem, termed Approximate Labeled Subtree Homeomorphism (ALSH), by introducing node-to-node similarity measures that are combined with the topological distance between the pattern and text to produce a single, comprehensive score expressing how close they are to each other (see Figure 1.2(b)). These findings, as well as additional combinatorial techniques that allow to further improve the time complexity of the defined problem, are described in Chapter 2.

Another important problem that is raised when computing the matching between the objects, is whether the found solution agrees with the expected one. More specifically, if we have some
prior knowledge on the matching to be found, how does the matching algorithm know about this? And if we could supply this knowledge as an additional input to the algorithm, can it improve the computation time? Chapter 3 describes the technique that utilizes the prior knowledge given as a set of matched pairs of nodes (termed seeds) in order to find a matching consistent with this prior knowledge and in order to improve the time complexity of the entire family of tree matching algorithms known as \textit{LCA-preserving mapping} algorithms (see definition in Chapter 3).

1.2 Applications and Suitable Similarity Measures

The main application considered in this thesis is the alignment (matching) of metabolic pathways. However, for the sake of clarity in presentation, before we discuss this main application, we would like to mention several other applications from the area of systems biology.

The first one is the comparison of RNA secondary structures. When we want to study the similarity between RNA molecules, it often becomes interesting to compare not only their sequences, but their structures as well. As two different sequences can provide similar secondary structures, comparison between such structures are necessary to lead to a better understanding of the functions of different RNA molecules. This secondary structure comparison can hence allow one to predict with better efficiency a spatial folding of a given molecule, to help in the identification of structural patterns that are preserved during the mutation process, and eventually rebuild phylogenetic trees.

A secondary structure can be represented by an ordered labeled tree \cite{124, 43}. Thus, secondary structure comparison (in which pseudo-knots are not taken into account) can be reduced to a tree comparison. The RNA structure can be decomposed in a way where the labels of the nodes represent either (1) structural elements (sequences of paired bases, hairpins, loops, bulges, stems, etc. or (2) nucleic acids (A, C, G, U), Watson-Crick pairs, Wobble pairs, etc. In that way considering only the structural patterns, this tree representation can be rough (situation (1)) of refined (situation (2)). Figure 1.3 represents an RNA structure, a rough tree representation of this structure, and a coding of the same structure by paired nucleotides. The nodes similarity measure can be either based on the topological elements resemblance or on the corresponding sequence alignment.

The second application is the comparison of phylogenetic (evolutionary) trees \cite{33}. The leaves of an evolutionary tree are always labeled with the taxa (or the corresponding gene/protein sequence), and permuting the labels on a tree with fixed topology generally produces a different evolutionary tree. Internal nodes which represent the hypothetical ancestors are generally unlabeled. Phylogenies may be rooted or unrooted, and edges may be weighted or unweighted. Order among descendants of each node is unimportant. For example, for a node in a rooted tree,
swapping the left and the right child does not change the tree.

Many biological problems involve the construction of a phylogenetic tree for some set of sequence data, and a variety of methods for inferring phylogenies are available. However, the choice of the phylogenetic reconstruction method can have a strong influence on the tree obtained for a given set of sequence data, both in terms of its topology and branch lengths. In addition, different gene trees can be obtained for some fixed set of species, where each gene tree is based on a different set of orthologous sequences chosen for the analysis. Comparison between gene trees and species trees can reveal consensus patterns of evolution as well as genes that diverge from this pattern. Also, comparing phylogenies allows revealing the regions where two trees agree or differ are therefore it is useful for assessing the quality of phylogenetic trees and analysis of different phylogenetic methods.

As mentioned above, our main application is the alignment of metabolic pathways. The collection of reactions and enzymes that an organism uses to achieve a certain metabolic function determines the architecture and topology of the pathway. Metabolic pathways can be abstracted either as metabolite graphs or as reaction (enzyme) graphs with specific graph-topological information, such as connectivity. In the first case, a metabolic pathway can be represented as a directed graph with substrates as vertices (nodes) and directed edges denoting reactions (labeled by enzymes) between the vertices. In the second case, which we use to model pathways in our application, enzymes denote vertices and metabolites denote edges. Two enzymes are related if they
activate reactions which share at least one chemical compound, either as substrate or as product. In the enzyme graph $G = (V, E)$ for a given pathway $P$, the vertex set $V$ consists of the enzymes present in the pathway $P$ and the set of edges $E$ represent the enzyme-enzyme relationships of the pathway. There exists a directed edge from enzyme $e_1$ to enzyme $e_2$ in $G$ if $e_1$ activates some reaction $A \to B$ (with substrate $A$ and product $B$) and $e_2$ activates some reaction $B \to C$ (with substrate $B$ and product $C$). Figure 1.4(b) and (c) illustrates both possible representations; note that the two are interchangeable.

We base our metabolic pathway alignment method on the subtree homeomorphism model for reasons that are both biologically and computationally driven. Biologically, a single enzyme in one pathway may replace a few consecutively acting enzymes in another pathway. The replacement can take place if the replacing enzyme is multifunctional and can thus catalyze several consecutive reactions, or if the enzyme uses an alternative catalysis that leads directly from the initial substrate to the final product. Note that enzymes that catalyze just a single reaction are more likely to be replaced than those that catalyze more reactions, for both biochemical and parsimony-related reasons. Translating this biological description into graph terms implies that degree-2 nodes may be deleted from the graph, a behavior which is perfectly captured by subtree homeomorphism. Computationally, the advantage of subtree homeomorphism over the more complex models (such as subgraph homeomorphism) is in that it has tractable solutions.
Chapter 4 describes a tool for alignment of metabolic pathways which is based on the algorithms presented in Chapter 2. As mentioned above, the core of this approach is node-to-node similarity measure, which should measure similarity between the enzymes. Possible similarity measures are described next.

1.2.1 Biological Similarity Measures

Several different measures of similarity between enzymes were developed recently based on the biological knowledge acquired so far by the community. For example, one of the first approaches was based on the observation that the similarity of the corresponding DNA or amino acids sequences, or structure similarity of the proteins conformation, implies a high likelihood of the two enzymes having the same biological function [2]. The main drawback of these methods is the high computational cost of comparing sequences or structures (see [42, 79, 113]), especially if they have to be performed as part of a more involved process. Moreover, they are not accurate enough in many contexts. Therefore, several alternative similarity measures have been developed recently. Here we describe a few of them:

**Enzyme Classification (EC).** Each of the enzymes that participate in a pathway is characterized by the reactions it catalyzes. The International Union of Biochemistry and Molecular Biology (IUBMB) has developed a classification scheme based on this observation. The scheme is hierarchical, with four levels. An EC number is represented as a string of four numbers separated by periods each corresponding to one level of the hierarchy. At the top of the hierarchy are six broad classes of enzymatic activity indicated by the first number of the EC number. The consecutive numbers indicate the sub-classes of the corresponding reaction. Based on the fact that the relationship between the proximity within the enzyme hierarchy and the reaction similarity is strong, we can measure the similarity between enzymes combining metabolic pathways based on the enzyme hierarchy. The similarity score is proportional to the number of leaves in the subtree rooted by the lowest common ancestor of two given enzymes. It is important to stress that the classification was carried out manually, without using any computational routine.

**Semantic similarity measure.** In many protein sequence databases, such as SWISS-PROT [10], the data is associated with a large amount of annotations. Those annotations range from semi-structured data, such as species information, to unstructured free text descriptions. In order to capture a common view, domain ontologies are used. An ontology provides a set of vocabulary terms that label domain concepts. The Gene Ontology (GO) [14] represents terms within a Directed Acyclic Graph (DAG) consisting of about 11000 terms, represented as nodes within the graph, connected by relationships, represented as edges. The ontological annotations can be used to measure the similarities in knowledge content or “semantic similarity” between entries in a data
resource (e.g. SWISS-PROT) [71]. This measure is based on GO’s notion of information content, which reflects the fact that the less frequently used terms are more informative. The information content is defined as the number of times each term, or any child term, occurs in the corpus (e.g. the SWISS-PROT database). This is expressed as probability. Given these probabilities, there are several measures of semantic similarity, but all of them use the information content of the shared parents of two terms.

**Functional similarity based on clustering.** Another way to express similarity is by using clustering techniques. Clustering may reveal facets of the relationship between protein/enzyme sequences that are not visible in sequence comparison since sequence similarity is not transitive, whereas homology (and biological function) is. The ProtoNet system [98] provides a hierarchical clustering of the protein space. The clustering is based primarily on an all-against-all BLAST [5, 4, 6] similarity test. With this similarity measure a sustained bottom-up clustering process is preformed by applying alternative rules for merging clusters. The outcome of this clustering process is a classification of the input proteins into a hierarchy of clusters of varying degrees of granularity. This clustering can be used for functional prediction, for defining superfamilies and subfamilies, and for similarity measure purposes.

### 1.2.2 Computational Similarity Measures

The main disadvantage of the biological similarity measures presented above is their rigidity. Despite of continuous updates, many enzymes remain unannotated or annotated to multiple functions. This causes much difficulties in aligning pathways consisting of many such enzymes.

Several recent studies attempted to establish computational measures for the similarity between genes that are based on the topological properties of metabolic networks. However, these approaches offer only a static description of the properties of interest and offer moderate (albeit significant) correlations with pertinent experimental data.

Using a constraint-based large-scale metabolic model, we developed two effectively computable measures of functional gene similarity, one based on the response of the metabolic network to gene knock-outs and the other based on the metabolic flux activity across a variety of growth media. We applied these measures to 750 genes comprising the metabolic network of the budding yeast. Comparing the *in silico* computed functional similarities to Gene Ontology (GO) annotations and gene expression data, we show that our computational method captures functional similarities between metabolic genes that go beyond those obtained by the topological analysis of metabolic networks alone, thus revealing dynamic characteristics of gene function. Interestingly, the measure based on the network response to different growth environments markedly outperforms the measure based on its response to gene knockouts, though both have some added synergistic value.
in depicting the functional relationships between metabolic genes. These findings are presented in Chapter 5.

1.3 Outline

This thesis is organized as follows: Chapter 2 formally defines the ALSH problem, provides computational background and reviews related work, describes basic algorithms for solving ALSH on different kinds of trees, and presents the computational techniques that allow to solve the problem efficiently. Then Chapter 3 describes another computational approach that utilizes prior knowledge about the matching to be found in order to improve the time complexity of a whole family of algorithms known as LCA-preserving mapping algorithms (that ALSH is a member of). Chapter 4 describes a method and a tool for the alignment of metabolic pathways that utilize a variant of the ALSH approach. Finally, Chapter 5 describes a novel approach for devising functional relations between metabolic genes that can be used also as a similarity measure between enzymes combining metabolic pathways. All of the described techniques have other important applications in systems biology as discussed in due course.

Appendix A provides the User’s Manual for the MetaPathwayHunter tool described in Chapter 4. Another related study of computational techniques that are capable of revealing regulatory mechanisms based on environmental input is presented in Appendix B. This study uses heuristics rather than exact optimization techniques in order to find the solution that maximizes the objective function, which is the number of co-expressed genes under different environmental conditions.
Chapter 2

Approximate Labeled Subtree Homeomorphism

In this chapter we address the challenge of extending exact labeled subtree homeomorphism to a new optimization problem by introducing node-to-node similarity measures that are combined with the topological distance between the pattern and text to produce a single, comprehensive score expressing how close they are to each other. More precisely, let $\Delta$ be a node-to-node similarity score table and $\delta$ denote a predefined (usually negative) score for deleting a node from a tree (Figure 2.1(b)).

**Definition 1 (ancestry preserving mapping)** A mapping $M$ of a tree $T_1 = (V_1, E_1)$ to a tree $T_2 = (V_2, E_2)$ is a bijection $M \subseteq V_1 \times V_2$ such that for all $(v_1, v_2), (w_1, w_2) \in M$, $v_1$ is an ancestor of $w_1$ in $T_1$ if and only if $v_2$ is an ancestor of $w_2$ in $T_2$.

We define the following similarity measure for two homeomorphic trees.

**Definition 2 (labeled subtree homeomorphism similarity score)** Let $T_1$ and $T_2$ be two labeled trees such that $T_2$ is homeomorphic to $T_1$, and let $M$ be a mapping from $T_1$ to $T_2$. The **labeled Subtree Homeomorphism (LSH) Similarity Score** of $M$, denoted $\text{LSH}(M)$, is

$$\text{LSH}(M) = \delta \cdot (|T_2| - |T_1|) + \sum_u \Delta[u, M(u)].$$

**Definition 3 (approximate labeled subtree homeomorphism problem)** The **Approximate Labeled Subtree Homeomorphism (ALSH) problem** is, given two undirected labeled trees $P$ and $T$, a scoring table which specifies the similarity scores between the label of any node appearing in $T$ and the label of any node appearing in $P$, and a predefined node deletion penalty, to find a mapping $M$ from $P$ to some subtree $t$ of $T$ such that $\text{LSH}(M)$ is maximal.
We start by describing how to compute (in a bottom-up fashion) alignments between $P$ and any homeomorphic subtree $t$ of $T$ which maximize the LSH score between $P$ and $t$. Our algorithms are based on the close relationship between subtree homeomorphism and maximum weight matchings in bipartite graphs. The ALSH problem is recursively translated into a collection of smaller ALSH problems, which are solved using maximum weight matching algorithms (Figure 2.2). (Simpler, maximum matching algorithms were applied in the algorithms for exact subtree morphisms [18, 41, 70, 107, 120].)

Our approach yields an $O(m^2n/\log m + mn \log n)$ algorithm for solving ALSH on unordered, unrooted trees, where $m$ and $n$ are the number of vertices in $P$ and $T$, respectively. Note that the time complexity of the exact subtree isomorphism/homeomorphism algorithms [107, 120], which do not take into account the node-to-node scores, is $O(m^{1.5}n/\log m)$. Thus, the enrichment of the model with the node similarity information only increases the time complexity by half an order. We also give an $O(mn)$ algorithm for the problem on rooted ordered trees and $O(mn \log m)$ and $O(mn)$ algorithms for unrooted cyclically ordered and unrooted linearly ordered trees respectively.

Also note that a related set of problems, where dynamic programming is combined with weighted bipartite matching, is that of finding the maximum agreement subtree and the maximum refinement subtree of a set of trees [57, 114]. Such algorithms utilize the special constraints imposed by the properties of evolutionary trees (internal nodes contain less information than leaves, the similarity assumption allows for scaling, etc.).
The rest of the chapter is organized as follows. Section 2.1 includes preliminaries, a basic $O(m^2n + mn \log n)$ time ALSH algorithm for rooted unordered trees, its extension to unrooted unordered trees without increasing the time complexity, and a basic $O(mn)$ ALSH algorithm for rooted ordered trees. Next, Section 2.2 presents a new $O(m^2n/\log m + mn \log n)$ solution for both rooted and unrooted unordered trees. Section 2.3 describes the $O(mn \log m)$ time ALSH algorithm for rooted cyclically ordered trees, and $O(mn)$ time ALSH algorithms for unrooted linearly ordered trees. Finally, conclusions and open problems are given in Section 2.4.

The results reported in this chapter appear in [87].

### 2.1 Basic Algorithms for ALSH

This section describes in brief the basic ALSH algorithms for unordered rooted and unrooted trees (Algorithm 1 and Algorithm 2, respectively) as well as for ordered rooted trees. The reader familiar with these basics can skip directly to Section 2.2. The presentation is given in summary form since some of the results appeared in my master’s thesis [96]. A more detailed presentation can also be found in [87].

#### 2.1.1 Rooted Unordered Trees

Let $T_r = (V_T, E_T, r)$ be the text tree which is rooted at $r$, and $P_{r'} = (V_P, E_P, r')$ be the pattern tree which is rooted at $r'$, respectively. For a node $v$ of $T_r$, let $t^v_r$ denote the subtree of $T_r$ that contains $v$ and all its descendents, and whose root is $v$. Similarly, we use $p^v_{r'}$ to denote a subtree of $P_{r'}$.

**Definition 4 (RScores table)** For each node $v \in V_T$ and for each node $u \in V_P$, $\text{RScores}[v, u]$ is the maximum LSH similarity score between $p^v_{r'}$ and some homeomorphic subtree of $t^v_r$, if such a subtree exists. Otherwise, $\text{RScores}[v, u]$ is $-\infty$.

The computation of $\text{RScores}[v, u]$ is performed recursively, in a postorder traversal of $T_r$. First, $\text{RScores}[v, u]$ is computed for every leaf node of $T_r$ and $P_{r'}$. Next, $\text{RScores}[v, u]$ is computed for each node $v \in V_T$ and $u \in V_P$, based on the values of the previously computed scores for the children of $v$ and $u$ as follows. Let $u$ be a node of $P_{r'}$ with children $x_1, \ldots, x_{c(u)}$ and $v$ be a node of $T_r$ with children $y_1, \ldots, y_{c(v)}$. After computing $\text{RScores}[y_j, x_i]$ for $i = 1, \ldots, c(u)$ and $j = 1, \ldots, c(v)$, a complete bipartite graph $G$ is constructed with bipartition $X$ and $Y$, where $X$ is the set of children of $u$, $Y$ is the set of children of $v$, and each node in $X$ is connected to every node in $Y$. An edge $(x_i, y_j)$ of $G$ is annotated with weight $\text{RScores}[y_j, x_i]$ (Figure 2.2).

$\text{RScores}[v, u]$ is then computed as the maximum between the following two terms:
Figure 2.2: The work done by Algorithm 1 during the computation of RScores\([v, u]\). The bipartite graph is constructed in order to compute the optimal weighted assignment between the children of \(u\) and those of \(v\). \(w_{ij}\) marks the optimal LSH similarity score for aligning the subtrees rooted at \(y_j \in T\) and \(x_i \in P\).

I. The node-to-node similarity value \(\Delta[v, u]\), plus the maximum weight of a matching in \(G\).

Note that this term is computed only if \(c(u) \leq c(v)\).

II. The maximum value among RScores\([y_1, u]\), …, RScores\([y_{c(v)}, u]\), plus \(\delta\) (the penalty for deleting \(v\)).

The optimal LSH similarity score, denoted \(\text{best\_score}\), is \(\text{best\_score} = \max_{j=1}^{n} \text{RScores}[y_j, x_m]\). Every node \(y_j \in T\) with RScores\([y_j, x_m]\) = \(\text{best\_score}\) is reported as a root of a subtree of \(T\) which bears maximal similarity to \(P\) under the LSH measure.

The pseudo-code of the algorithm described above, denoted Algorithm 1, is given on Page 15.

**Time Complexity Analysis**

**Theorem 1** Algorithm 1 computes the optimal ALSH solution for two rooted unordered trees in \(O(m^2n + mn \log n)\) time. Moreover, under the similarity assumption, the algorithm has an \(O(m^{1.5}n \log (nC))\) time complexity.

**Proof** The time complexity analysis of the algorithm is based on the following observation.

**Observation 1** \(\sum_{u=1}^{m} c(u) = m - 1\) and \(\sum_{v=1}^{n} c(v) = n - 1\).

14
Algorithm 1: ALSH for Rooted Unordered Trees

**Input:** Rooted trees $T = (V_T, E_T, r)$ and $P = (V_P, E_P, r')$.

**Output:** The root of the subtree $t$ of $T$ that has the highest similarity score to $P$, if $T$ has a subtree which is homeomorphic to $P$.

1: procedure RootedUnorderedALSH
2:   for each node $u$ of $P$ in postorder do
3:     for each node $v$ of $T$ in postorder do
4:       if $u$ is leaf then
5:         if $v$ is leaf then
6:           $\text{RScores}[v, u] \leftarrow \Delta[v, u]$
7:         else
8:           $\text{RScores}[v, u] \leftarrow \text{ComputeScoresForTextNode}(v, u)$
9:         end if
10:     else
11:       if $\text{Level}(u) > \text{Level}(v)$ then
12:         $\text{RScores}[v, u] \leftarrow -\infty$
13:       else
14:         $\text{RScores}[v, u] \leftarrow \text{ComputeScoresForTextNode}(v, u)$
15:       end if
16:     end if
17:   end for
18: end for
19: end procedure

20: procedure ComputeScoresForTextNode($v, u$)
21:   Let $k$ denote the out-degree of node $u$ and $\ell$ denote the out-degree of node $v$.
22:   if $k > \ell$ then
23:     $\text{AssignmentScore} \leftarrow -\infty$
24:   else
25:     Construct a bipartite graph $G$ with node bipartition $X$ and $Y$ such that
26:     $X = \{x_1, \ldots, x_k\}$ is the set of children of $u$, $Y = \{y_1, \ldots, y_\ell\}$ is the set of
27:     children of $v$, and every node $u_i \in X$ is connected to every node $v_j \in Y$
28:     via an edge whose weight is $\text{RScores}[v_j, u_i]$.
29:     Set $\text{AssignmentScore}$ to the maximum weight of a matching in $G$.
30:   end if
31:   $\text{BestChild} \leftarrow \max_{j=1}^\ell \text{RScores}[y_j, u]$
32:   return $\max\{\Delta[v, u] + \text{AssignmentScore}, \text{BestChild} + \delta\}$
33: end procedure
Algorithm 1 calls procedure ComputeScoresForTextNode once for each node pair \((v \in T, u \in P)\). The dominant part of this procedure’s work is spent in computing a maximum weight matching in a complete bipartite graph with \(c(v) + c(u)\) nodes and \(c(v) \cdot c(u)\) edges. Therefore, using Fredman and Tarjan’s algorithm [36], each maximum weight matching computation takes \(O(c(u)^2 \cdot c(v) + c(u) \cdot c(v) \log c(v))\) time.

Summing up the work over all node pairs we get

\[
O(\sum_{u=1}^{m} \sum_{v=1}^{n} (c(u)^2 c(v) + c(u)c(v) \log c(v))) \overset{\text{Obs. 1}}{=} O(\sum_{u=1}^{m} c(u)^2 n + c(u)n \log n) \overset{\text{Obs. 1}}{=} O(m^2 n + mn \log n).
\]

Under the similarity assumption (namely, all scores assigned to the edges of \(G\) are integers in the range \([-C, \ldots, C]\), where \(C = O(n^k)\) for some constant \(k\)), the algorithm can be modified (Step 6 of procedure ComputeScoresForTextNode) to run in time \(O(m^{1.5} n \log(nC))\) by employing the algorithm of Gabow and Tarjan [37] for weighted bipartite matching.

### 2.1.2 Unrooted Unordered Trees

Let \(T = (V_T, E_T)\) and \(P = (V_P, E_P)\) be two unrooted trees. The ALSH problem on \(P\) and \(T\) can be solved in a naïve manner as follows. Select an arbitrary node \(r\) of \(T\) to get the rooted tree \(T^r\). Next, for each \(u \in P\) solve the rooted ALSH problem on \(P^u\) and \(T^r\). This method yields an \(O(m^3 n + m^2 n \log n)\) strongly polynomial time algorithm for ALSH on unrooted unordered trees, and an \(O(m^{2.5} n \log(nC))\) time algorithm under the similarity assumption with an integrality cost function restriction. These bounds can be reduced back to \(O(m^2 n + mn \log n)\) and \(O(m^{1.5} n \log(nC))\) time, respectively, by utilizing the decremental properties of the maximum weight matching algorithm. Again, for a more detailed presentation see [96, 87].

The algorithm starts by selecting a vertex \(r\) of \(T\) to be the designated root. \(T^r\) is then traversed in postorder, and each internal vertex \(v \in T^r\) is compared with each vertex \(u \in P\). Let \(y_1, \ldots, y_{c(v)}\) be the children of \(v \in T^r\), and let \(x_1, \ldots, x_{d(u)}\) be the neighbors of \(u \in P\). When computing the score for the comparison of \(u\) and \(v\) we need to take into account the fact that since \(P\) is unrooted, each of the neighbors of \(u\) may serve as a parent of \(u\) in a mapping of \(P\) to some subtree \(t_u^r\). Suppose node \(x_i \in P\), which is a neighbor of \(u \in P\), serves as the parent of \(u\) in one of these subtree alignments. Since \(x_i\) is the chosen parent, the children set of \(u\) includes all its neighbors but \(x_i\). Therefore, the computation of the similarity score for \(v\) versus \(u\) requires computing the maximum weight matching between the children of \(v\) and all neighbors of \(u\) except \(x_i\). Since each neighbor \(x_i\) of \(u\) is a potential parent of a subtree rooted in \(u\), our algorithm will actually need to compute the maximum weight matching in a series of graphs \(G_i = (X_i \cup Y, E)\) for \(i = 1, \ldots, d(u)\), where \(Y\) consists of all the children of \(v\) in \(T^r\), while \(X_i\) consists of all the
neighbors of \( u \) except \( x_i \). Therefore, we define the table UScores as follows.

**Definition 5 (UScores table)** For every vertex \( v \in T^r \), every vertex \( u \in P \), and every vertex \( x_i \in \text{neighbors}(u) \), \( \text{UScores}[v, u, x_i] \) is the maximum LSH similarity score between \( p_{x_i}^u \) and some homeomorphic subtree of \( t_v^r \), if such subtrees exist. Otherwise, \( \text{UScores}[v, u, x_i] = -\infty \). Moreover, \( \text{UScores}[v, u, \phi] \) is the maximum LSH similarity score between \( P^u \) and some homeomorphic subtree of \( t_v^r \), if one exists.

The computation of \( \text{UScores}[v, u, x_i] \) is carried out as follows. \( \text{UScores}[v, u, x_i] \) is set to the maximum between the following two terms:

I. The node-to-node similarity value \( \Delta[v, u] \), plus the maximum weight of a matching in \( G_i \).

   Note that this term is only computed if \( d(u) - 1 \leq c(v) \).

II. The maximum value among \( \text{UScores}[y_1, u, x_i], \ldots, \text{UScores}[y_{c(v)}, u, x_i] \) plus \( \delta \).

The maximal LSH similarity score between \( P \) and \( T \) is \( \text{best\_score} = \max_{v \in V_T, u \in V_P} \text{UScores}[v, u, \phi] \). Every node pair \( (j \in T, i \in P) \) for which \( \text{UScores}[j, i, \phi] = \text{best\_score} \) is reported.

The pseudo-code of the above algorithm, denoted Algorithm 2, is given on Page 18.

**Time Complexity Analysis**

The time complexity bottleneck of Algorithm 2 is in the need to compute, for each pair \( u \in P \) and \( v \in T \), maximum weight matchings in a series of bipartite graphs \( G_i = (X_i \cup Y, E) \) for \( i = 1, \ldots, d(u) \). This is in contrast to Algorithm 1, where only one maximum weight matching is computed for each pair \( u \in P \) and \( v \in T \). However, the next lemma shows that the decremental nature of weighted bipartite matching makes it possible to compute the matchings for \( G_1, \ldots, G_{d(u)} \) in the same asymptotic time complexity as the maximum weight matching computation for the single graph \( G = (X \cup Y, E) \).

For the following lemmas, denote \( k = |X| \) and \( \ell = |Y| \). Note that \( k \leq \ell \).

Also we use the following observation.

**Observation 2** The sum of vertex degrees in an unrooted tree \( P \) is \( \sum_{u=1}^{m} d(u) = 2m - 2 \).

**Lemma 1** Given a maximum weight matching \( M \) of \( G \), a maximum weight matching of \( G_i \) can be computed via one run of a single-source shortest-path algorithm.

**Lemma 2** Computing the maximum weight matchings for the bipartite graphs \( G_1, \ldots, G_k \) can be done in \( O(k^2\ell + k\ell \log \ell) \) time. Moreover, under the similarity assumption, computing these matchings can be done in \( O(k^{1.5}\ell \log(\ell C)) \) time.
Algorithm 2 ALSH for Unrooted Unordered Trees

Input: Unrooted trees $T = (V_T, E_T)$ and $P = (V_P, E_P)$.

Output: The root of the subtree $t$ of $T$ which has the highest similarity score to $P$, if $T$ has a subtree which is homeomorphic to $P$.

1: procedure UnrootedUnorderedALSH
2: Pick a vertex $r$ of $T$ to be the root of $T$
3: for all $u \in P$, $v \in T$, $x_i \in P$ do
4: \hspace{1em} UScores[v, u, x_i] \leftarrow -\infty
5: end for
6: for each leaf $v$ of $T_r$ do
7: \hspace{1em} for each leaf $u$ of $P$ do
8: \hspace{2em} UScores[v, u, parent(u)] \leftarrow \Delta[v, u]
9: \hspace{1em} end for
10: end for
11: for each internal node $v$ of $T$ in postorder do
12: \hspace{1em} ComputeScoresForTextNode($v$)
13: \hspace{1em} end for
14: \hspace{1em} best_score \leftarrow \max_{i=1, \ldots, m, j=1, \ldots, n} UScores[v_j, u_i, \phi]
15: end procedure

16: procedure ComputeScoresForTextNode($v$)
17: \hspace{1em} for each node $u$ of $P$ do
18: \hspace{2em} BestChild($v, u, x_i$) \leftarrow \max_{j=1, \ldots, n} UScores[y_j, u, x_i]
19: \hspace{1em} end for
20: Construct a bipartite graph $G$ with node bipartition $X$ and $Y$ such that
21: $X = \{x_1, \ldots, x_k\}$ is the set of neighbors of $u$, $Y = \{y_1, \ldots, y\ell\}$ is the set of
22: children of $v$, and every node $x_i \in X$ is connected to every node $y_j \in Y$
23: via an edge whose weight is $UScores[y_j, x_i, u]$
24: Let $X_0 = X$ and $X_i = X - \{x_i\}$
25: \hspace{1em} for all $1 \leq i \leq k$ do
26: \hspace{2em} if Level($v$) < Level($u$, $x_i$) then
27: \hspace{3em} UScores[$u, v, x_i$] \leftarrow -\infty
28: \hspace{2em} \hspace{1em} else
29: \hspace{3em} \hspace{1em} if $k > \ell + 1$ then
30: \hspace{4em} AssignScore($X_i, Y$) \leftarrow -\infty
31: \hspace{3em} \hspace{1em} \hspace{1em} else
32: \hspace{4em} \hspace{1em} AssignScore($X_i, Y$) \leftarrow maximum weight matching score in $G_i$
33: \hspace{3em} \hspace{1em} end if
34: \hspace{3em} UScores[$v, u, x_i$] \leftarrow \max\{\Delta[v, u] + AssignScore($X_i, Y$), BestChild($v, u, x_i$, $\phi$)\}
35: \hspace{2em} \hspace{1em} end if
36: \hspace{2em} end for
37: \hspace{1em} if $k > l$ then
38: \hspace{2em} UScores[$v, u, \phi$] \leftarrow -\infty
39: \hspace{1em} \hspace{1em} else
40: \hspace{2em} \hspace{1em} UScores[$v, u, \phi$] \leftarrow $\Delta[v, u]$ + maximum weight matching in $G$
41: \hspace{2em} \hspace{1em} end if
42: \hspace{1em} end for
43: end procedure
Theorem 2 Algorithm 2 computes the optimal ALSH solution for two unrooted unordered trees in $O\left(m^2n+mn\log n\right)$ time. Under the similarity assumption, the time complexity of the algorithm is $O(m^{1.5}n\log(nC))$.

The proofs of the lemmas and the theorem along with a more detailed explanations can be found in [87].

2.1.3 Ordered Rooted Trees

A simplified version of Algorithm 1 can be used to solve the ALSH problem on ordered rooted trees. The simplification is based on the fact that the assignment problems now turn into maximum weight matching problems on ordered bipartite graphs, where no two edges are allowed to cross each other in the matchings. (We refer the reader to [120] for a discussion of non-crossing plain bipartite matching in the context of exact ordered tree homeomorphism.) Also note that, in the context of our ALSH solutions, we apply a rectangular case of the perfect assignment problem to the graph $G = (X \cup Y, E)$, i.e. $|X| \leq |Y|$ and all nodes in $X$ must eventually be paired with some node in $Y$. Therefore, the assignment computation reduces to the Approximate Weighted Episode Matching optimization problem, as defined below.

Definition 6 (approximate weighted episode matching problem) The Approximate Weighted Episode Matching Problem is: given a pattern string $X$, a source string $Y$, and a character-to-character similarity table $\Delta[\Sigma_X, \Sigma_Y]$, find among all $|X|$-sized subsequences of $Y$ a subsequence $Q$ which is most similar to $X$ under $\Delta$ (i.e. a subsequence $Q$ that maximizes the sum $\sum_{i=1}^{|X|} \Delta[Q_i, X_i]$).

Lemma 3 The Approximate Weighted Episode Matching problem can be solved in $O(|X| \cdot |Y|)$ time.

Proof The matching problem is solved by applying the classical dynamic programming string alignment algorithm on a $|X| + 1$ rows by $|Y| + 1$ columns graph (Figure 2.3(a)). All horizontal edges in the graph, corresponding to character deletions from $Y$, are assigned a score of zero. All vertical edges in the graph, corresponding to character deletions from $X$, are assigned a score of $-\infty$. A diagonal edge leading into vertex $(x_i, y_j)$ corresponds to the string-edit operation of substituting the $i$th character of $X$ with the $j$th character of $Y$, and is therefore assigned the score $\Delta[i, j]$. During the initialization stage, the scores of all vertices in the first row of the dynamic programming graph are set to zero. The dynamic programming algorithm is then applied to this alignment graph.
When the algorithm completes, all vertices in the last row of the graph are scanned for the best score. Any vertex in the last row which carries the optimal score represents a highest scoring approximate weighted episode of $X$ in $Y$.

Clearly, after running the algorithm described in the proof of Lemma 3 on the alignment graph constructed for strings $X$ and $Y$, a highest-scoring path $P$ in the alignment graph will correspond to a maximum weight matching $M$ in the corresponding ordered bipartite graph $G = (X \cup Y, E)$, where no two edges in the matching are allowed to cross. A diagonal edge in $P$ connecting vertex $(x_i - 1, y_j - 1)$ with vertex $(x_i, y_j)$ corresponds to a pairing of $x_i$ and $y_j$ in $M$.

**Time Complexity Analysis**

**Theorem 3** The ALSH problem on rooted ordered trees can be solved in $O(mn)$ time.

**Proof** By Lemma 3, the time complexity is

$$
\sum_{v=1}^{n} \sum_{u=1}^{m} O(c(v) \cdot c(u)) \overset{\text{Obs. 1}}{=} O(1) \sum_{v=1}^{n} O(m \cdot c(v)) \overset{\text{Obs. 1}}{=} O(mn).
$$

**2.2 A More Efficient ALSH Algorithm for Unordered Trees**

In this section we show how the dominant term in the time complexity of the algorithm described in the previous section can be reduced by a log $m$ factor, assuming a constant-sized label alphabet. We will use the previous algorithm but solve the matching problems more efficiently by employing the notion of a clique partition of a bipartite graph [30, 107]. The modified algorithm is the same
as Algorithm 2 with the exception that, when computing \( UScores[v, u, x_i] \) between two internal nodes \( v \) and \( u \) (in Step 12 of procedure ComputeScoresForTextNode), we solve the assignment problem differently.

Let \( v \) be some vertex in \( T' \) whose children are \( y_1, \ldots, y_{c(v)} \) and let \( u \) be a vertex in \( P \) whose neighbors are \( x_1, \ldots, x_{d(u)} \). Denote \( X = \{x_1, \ldots, x_{d(u)}\} \) and \( Y = \{y_1, \ldots, y_{c(v)}\} \). Let \( G \) be the corresponding complete bipartite graph on \( X \cup Y \). We compute a maximum weight matching in \( G \) as follows. Let \( N(x_i) \) be the sequence of the weights of the edges \([x_i, y_1], \ldots, [x_i, y_{c(v)}]\). We sort the vertices of \( X \) where the key of a vertex \( x \) is \( N(x) \). Afterwards, we split \( X \) into sets of equal keys \( X_1, X_2, \ldots, X_{clusters_u} \) (i.e., for every two vertices \( x, x' \in X^i \), every edge \([x, y] \) has the same cost as the edge \([x', y]\)). Note that every set \( X^i \cup Y \) induces a complete bipartite subgraph of \( G \).

We next build a network \( G^* \) whose vertices are \( V^* = X \cup Y \cup \{c_1, \ldots, c_{clusters_u}, s, t\} \). The edges are \( E^* = E_1 \cup E_2 \cup E_3 \) where \( E_1 = \{[s, x_i] : x_i \in X\} \cup \{[y_i, t] : y_i \in Y\} \), \( E_2 = \{[x_i, c_j] : j \leq clusters_u, x_i \in X^j\} \), and \( E_3 = \{[c_j, y_i] : j \leq clusters_u, y_i \in Y\} \) (see Figure 2.4.A) . All edges have capacity 1. Edges from sets \( E_1 \) and \( E_2 \) are assigned a cost of zero. An edge of type \( E_3 \) from \( c_j \) to \( y_i \) is assigned a cost which is identical to the cost of the edge \([x, y_i]\) where \( x \) is any vertex belonging to the set \( X^j \). The source is \( s \) and the sink is \( t \). We find a min-cost max-flow \( f^* \) in \( G^* \) and construct from this flow a maximum weight matching in \( G \).

Figure 2.4: (a) Graph \( G' \) with a bipartition of its vertices into sets \( X \) and \( Y \), and the compressed graph \( G^* \) that contains an additional layer \( C \). (b) The mapping \( M \) of subunits of sub-paths from \( G^* \) into subunits of sub-paths in \( G' \).
Let $G'$ be the network that corresponds to $G$ and defined as follows: Let $s, t$ be two new vertices that do not appear in $G$. Construct a graph $G'$ with vertex set $V \cup \{s, t\}$, source $s$, sink $t$, and capacity-one edges: an edge $[s, x]$ of cost zero for every $x \in X$, an edge $[y, t]$ of cost zero for every $y \in Y$, and an edge $[x, y]$ of cost $-w(x, y)$ for every $[x, y] \in E$. An integral flow $f$ on $G'$ defines a matching on $G$ of size $|f|$ and weight $-\text{cost}(f)$ given by the set of edges $[x, y]$ such that $[x, y]$ has one unit of flow.

Lemma 4 ([25], [34], [115]) A flow $f$ is minimum cost iff its residual graph $R$ has no negative cost cycle.

Lemma 5 A min-cost max-flow in $G'$ can be found by finding a min-cost max-flow in $G^*$.

Proof We define a mapping $\mathcal{M}$ that maps paths in $G^*$ to paths in $G'$. The mapping $\mathcal{M}$ is by partitioning sub-paths in $G^*$ into subunits which are then mapped onto their corresponding subunits in $G'$. Consider the 6 cases of sub-unit mappings shown in Figure 2.4. Note that this mapping preserves the endpoints and the costs of each subunit except the subunit of Type 4 which does not have a mapping. Thus we can partition a path in $G^*$ into subunits, map each subunit into subunit in $G'$, and then reconstruct the path in $G'$ with exactly the same cost and the same endpoints. A subunit of Type 4 can be omitted during the mapping since it donates only 0 cost to the pathway in which it participates.

Following the algorithm of [19] for computing minimum cost maximum flow, a series of $d(u)$ stages is executed, where at each stage a new flow $f^*_i$ is computed which is an increment by one of the previous flow. Each incremental step is realized by computing the minimum cost path from $s$ to $t$ and infusing one flow unit along this path. Using the mapping $\mathcal{M}$ as demonstrated in Figure 2.4, each such increment path in the graph $G^*$ can be mapped into a corresponding minimum cost increment path in $G'$. Therefore, in any stage $i$ of the min-cost max-flow computation on $G^*$, $f^*_i$ is equal in cost and value to the corresponding flow $f_i$ in $G'$, as induced by $\mathcal{M}$. Clearly, any cycle in $G^*$ is mapped by $\mathcal{M}$ onto a cycle in $G'$ with exactly the same cost. Thus, if there is a negative cost cycle in the residual graph of $G^*$ at any stage of the min-cost max-flow computation, then it immediately follows that there is a corresponding negative cost cycle in the residual graph of $G'$. According to Lemma 4, if there is a negative cost cycle in the residual graph of $G'$, then the flow is not minimum cost. Thus if there is a negative cost cycle in a residual graph of $G^*$ then the corresponding flow in graph $G'$ is not minimal, in contradiction to [19].

We denote by $D(u)$ the number of distinct trees in the forest $p_{x1}^u, \ldots, p_{xd(u)}^u$.

Lemma 6 The maximum weight matching in the bipartite graph corresponding to $u \in P$ and $v \in T$ can be computed in $O(d(u) \cdot (D(u)c(v) + c(v) \log c(v)))$ time.
Proof If some pair of rooted trees \( p^u_{x_i} \) and \( p^u_{x_j} \) are isomorphic, then in the graph \( G \) the vertices \( x_i \) and \( x_j \) have exactly the same neighbors, and this remains true in \( G^* \). Therefore clusters \( u \leq D(u) \). The time for constructing \( G \) is \( O(d(u)c(v)) \) and the time for constructing \( G^* \) is \( O(d(u)c(v)\log c(v)) \).

We now bound the time for finding a min-cost max-flow in \( G^* \): The size of \( E_1 \) is at most \( |X| + |Y| = d(u) + c(v) \). The size of \( E_2 \) is \( |X| = d(u) \) and the size of \( E_3 \) is \( |\text{clusters}_u \cdot c(v)| \). Hence, the number of edges in \( G^* \) is \( O(\text{clusters}_u \cdot c(v)) \). The number of vertices in \( G^* \) is at most \( d(u) + c(v) + \text{clusters}_u + 2 = O(c(v)) \), since \( d(u) - 1 \leq c(v) \). Now, the maximum weight matching algorithm performs \( d(u) \) stages, and each stage takes \( O(E^* + V^* \log V^*) \) time. Hence, the total time is \( O(d(u) \cdot (\text{clusters}_u \cdot c(v) + c(v) \log c(v))) = O(d(u) \cdot (D(u)c(v) + c(v) \log c(v))) \) time.

**Time Complexity Analysis**

By Lemma 6, the total time complexity is

\[
O\left(\sum_{u=1}^{m} \sum_{v=1}^{n} (d(u)(\text{clusters}_u \cdot c(v)) + c(v) \log c(v)) \right) \text{Obs.} 1 = O\left(\sum_{u=1}^{m} d(u) \cdot \text{clusters}_u \cdot n + d(u)n \log n \right)
\]

\[
\text{Obs.} 2 = O(n \sum_{u=1}^{m} d(u)D(u) + mn \log n).
\]

We next turn to bound the summation \( O(\sum_{u=1}^{m} d(u)D(u)) \). As a preliminary step, we set an asymptotically tight bound on the number of distinct labeled rooted trees in a forest of \( n \) vertices.

**Lemma 7** Let \( f(n) \) be the number of distinct labeled rooted trees in a forest of \( n \) vertices. Assuming constant label alphabet, \( f(n) = O(n/\log n) \).

**Proof** We will prove that for any forest with \( n \) vertices labeled by an alphabet of size \( \sigma \), \( f(n) = O(n \log \sigma/\log n) \). It is known that the number of distinct rooted unlabeled trees with \( i \) vertices is at most \( c^i \) for some constant \( c \). Therefore, the number of distinct rooted labeled trees with \( i \) vertices is at most \( (\sigma c)^i \).

If we have a forest of labeled rooted trees and \( r_i \) is the number of trees with \( i \) vertices, then the number of distinct trees in this forest is at most \( \sum_{i=1}^{n} \min(r_i, (\sigma c)^i) \). Hence,

\[
f(n) \leq \max \left\{ \sum_{i=1}^{n} \min(r_i, (\sigma c)^i) : r_1, \ldots, r_n \in \mathbb{N}, \sum_{i=1}^{n} ir_i \leq n \right\}
\]

Let \( x \) be the minimum integer for which \( \sum_{i=1}^{x} i(\sigma c)^i \geq n \). Let \( r_1, \ldots, r_n \) be the integers that maximize \( \sum_{i=1}^{n} \min(r_i, (\sigma c)^i) \) under the constraint \( \sum_{i=1}^{n} ir_i \leq n \). Using the arguments of [107], we can assume that \( r_i \leq (\sigma c)^i \) for all \( i \) and \( r_i = 0 \) for all \( i > x \). Therefore, \( f(n) \leq \sum_{i=1}^{x} (\sigma c)^i = O((\sigma c)^x) \). The lemma follows from the fact that \( x = \log_{\sigma c} n - \log_{\sigma c} \log_{\sigma c} n - O(1) \).

23
Lemma 8 \( \sum_{u=1}^{m} d(u)D(u) = O(m^2/ \log m) \).

**Proof** Let \( \epsilon \) denote some constant \( 0 < \epsilon < 1/2 \). We call a vertex of \( P \) heavy if its degree is at least \( 2m^{1-\epsilon} \), otherwise the vertex is called light. Clearly, the number of heavy vertices is at most \( m' \). If \( u \) is a heavy vertex and \( w \) is a neighbor of \( u \) such that \( P_u^w \) does not contain a heavy vertex, then we call every vertex in \( P_u^w \) a private vertex of \( u \). We denote by \( \ell_v \) the number of the private vertices of a heavy vertex of \( v \).

We split \( \sum_{u=1}^{m} d(u)D(u) \) into two sums. Summing over the light vertices of \( P \) we have

\[
\sum_{v \in P \text{ is light}} d(v)D(v) \leq \sum_{v \in P \text{ is light}} d(v)^2 \leq 2m^{1-\epsilon} \cdot \sum_{v \in P \text{ is light}} d(v) \leq O(2m^{1-\epsilon}2m = 2^2m^{2-\epsilon}).
\]

Shamir and Tsur [107] proved that for any heavy vertex \( v \), \( d(v) \leq m^\epsilon + \ell_v \) and \( D(v) \leq m^\epsilon + f(\ell_v) \). Therefore, summing over the heavy vertices we have

\[
\sum_{v \in P \text{ is heavy}} d(v)D(v) \leq \sum_{v \in P \text{ is heavy}} (m^\epsilon + \ell_v)(m^\epsilon + f(\ell_v)).
\]

Since \( f(\ell_v) \leq \ell_v \leq m \) and since the number of heavy vertices is at most \( m' \), the last sum is bounded by

\[
\sum_{v \in P \text{ is heavy}} (m^\epsilon + m^{1+\epsilon} + m^{1+\epsilon} + \ell_v(f(\ell_v))) \leq 3m^{1+2\epsilon} + \sum_{v \in P \text{ is heavy}} \ell_v f(\ell_v).
\]

By Lemma 7, the fact that \( \sum\limits_{v \in P \text{ is heavy}} \ell_v \leq m \) (as each vertex can be a private vertex of at most one heavy vertex), and the fact that the function \( h(x) = x^2/\log x \) is convex, we obtain

\[
\sum_{u=1}^{m} d(u)D(u) \leq 3m^{1+2\epsilon} + \sum_{v \in P \text{ is heavy}} c\ell_v^2/\log \ell_v \leq 3m^{1+2\epsilon} + cm^2/\log m = O(m^2/ \log m). \quad \blacksquare
\]

We have thus proved the following theorem.

**Theorem 4** The described algorithm computes the optimal ALSH solution for two unrooted unordered trees in \( O(m^2n/ \log m + mn \log n) \) time.

### 2.3 An ALSH Algorithm for Ordered Unrooted Trees

In this section we extend the algorithm for ordered rooted trees to apply to unrooted trees. There are two standard ways to order unrooted trees: the cyclic order and the linear one.
A linearly ordered set $X$ of children of a node $u$ is a poset $(X, \leq)$ (a partial order relation on $X$) which has the property of comparability:

for all $x, y \in X$, either $x \leq y$ or $y \leq x$.

The binary relation $\leq$ is then called a linear ordering.

A cyclic order on a set $X$ of the children of node $u$ with $k$ elements is an arrangement of $X$ as on a clock face, for a $k$-hour clock. That is, rather than an order relation on $X$, we define on $X$ only the functions “element immediately before” and “element immediately following” for any given $x_i$, in such a way that activating one of these functions cycles once through the elements as $x_1, x_2, \ldots, x_k$.

Both orders are illustrated in Figure 2.5.

![Figure 2.5: Cyclic and linear orders of unrooted trees.](image)

### 2.3.1 Cyclically Ordered Unrooted Trees.

The problem of ALSH on cyclically ordered unrooted trees can be reduced to the **Cyclic String Comparison** problem, defined as follows [74]: Given two strings $P$ and $T$, find a cyclic shift of $P$ which can be best aligned with $T$. More precisely, let $T = T_1 \ldots T_\ell$ and $P = P_1 \ldots P_k$. We wish to find an index $i$ for which the optimal alignment of $P_i P_{i+1} \ldots P_k P_1 \ldots P_{i-1}$ with $T$ gives the best score. An $O(k \ell \log k)$ algorithm for the Cyclic String Comparison problem, which applies to general weights scoring schemes, was described in [74]. An $O(k \ell)$ algorithm which applies to rational numbers scoring schemes was given in [101].

An efficient solution to ALSH on Cyclically Ordered Unrooted Trees can be obtained by applying the following reduction. Consider computing the $(2k + 1) \times (\ell + 1)$ grid graph for the comparison of the string $PP$ ($P$ concatenated with $P$) versus $T$ (see Figure 2.6(left)). Each cyclic shift of $P$ defines a starting point in the source area for the alignment path to the appropriate
point in the destination area. The alignment scores can be computed efficiently in an incremental manner using the cyclic string comparison algorithms of [74, 101]. Our problem is a variant of this problem where in each increment step another neighbor of node \( u \) serves as its parent and thus cannot participate in the alignment. Therefore, the sliding window in our problem is of size \( k - 1 \) instead of \( k \) (see Figure 2.6(right)).

![Figure 2.6: The cyclic string comparison problem.](image)

**Time Complexity Analysis.** The time complexity for each pair of vertices is \( O(kl \log k) \) using the algorithm of Maez [74] or \( O(kl) \) for a rational number scoring scheme using the algorithm of Schmidt [101]. Therefore, summing up the work over pairs of vertices yields a total time complexity of \( O(mn \log m) \) and \( O(mn) \), respectively.

### 2.3.2 Linearly Ordered Unrooted Trees.

As mentioned above, the naïve extension of the rooted case requires an additional order in the time complexity since each of the neighbors of node \( u \) in the pattern tree eventually could serve as the parent of \( u \) and will not participate in the children’s matching for this computation. In this section we use the decomposition technique for the linearly ordered case to reduce the complexity to the bounds of the rooted case.

Our technique is based on Hirschberg’s algorithm for the linear space sequence alignment [47]. Similarly to Hirschberg, we calculate the forward “alignment” using the algorithm for the rooted ordered trees; the same technique applies to the backward “alignment”, which is the same as the forward alignment but in the opposite direction. Then we throw out the line \( i \), which represents
the current father of node \( u \) and stitch the forward and backward “alignments” by finding the best stitching point (see Figure 2.3(b)):

\[
stitching\_point = \max\{F[i - 1, j] + B[i + 1, j + 1]\} \quad \text{for} \; j = 0, \ldots, n - 1.
\]

Clearly, the value \( F[i - 1, j] \) represents the best alignment between \( X[0 \ldots i - 1] \) and \( Y[0 \ldots j] \). The value \( B[i + 1, j + 1] \) represents the best alignment between \( X[m \ldots i + 1] \) and \( Y[n \ldots j + 1] \). Therefore, the maximal sum corresponds to an optimal alignment of \( X - x_i \) and \( Y \).

**Time Complexity Analysis.** The calculations of both forward and backward alignments requires \( O(mn) \) time. The calculation of the stitching point requires \( O(n) \) time. Therefore, the overall time complexity is \( O(mn) \).

### 2.4 Discussion

Motivated by real life scenarios from a variety of domains, such as bioinformatics, semistructured databases and natural language processing, we have extended the problem of homeomorphic pattern matching in trees to reflect both topological resemblance as well as node labels’ similarity (rather than strict equality) so as to form one comprehensive framework. We solved the challenges presented by our extended formulation, and succeeded in restricting the additional overhead in time complexity to half an order. Future research directions may include applying the idea of comparison enrichment by similarity scores to the more advanced tree inclusion problem, as well as to more complex topologies such as series-parallel graphs, graphs with limited tree-width, and directed acyclic graphs.

Still, in real life applications, the tree-restricted topologies might be too strong a constraint. For example, metabolic pathways can be modeled by graphs that are very close to trees but still are not trees. In such cases a slightly changed version of the described algorithms can be used after performing certain preprocessing steps. These findings are described in Chapter 4 as a part of a computational tool for the alignment of metabolic pathways.

Another important question is the consistency of the resulting matching with the expected one. Chapter 3 discusses this issue in depth.
Chapter 3

Seeded Tree Alignment

Two main issues arise during the analysis of every algorithmic approach: correctness and time complexity. When we use an algorithm as a black box, the correctness of the obtained result is the outcome of the applied algorithm. And this leads us to the question of consistency: does the obtained solution agree with the expected one? Specifically, does the resulting matching (alignment) agree with the surmised outcome? In many cases there is some prior knowledge about the matching to be found, which is not included in the set of constraints of the applied algorithm and thus the obtained solution may disagree with the expected one. In this section we describe a technique that uses this prior knowledge to find the solution that is consistent with the provided data. Moreover, since the presented approach can be viewed as a constrained optimization problem, we can address the second issue and improve the time complexity of an entire family of algorithms known as LCA-preserving matching algorithms (the family that ALSH belongs to).

We define prior knowledge as a set of paired nodes \((u_i, v_j)\) where \(u_i \in T_1\) and \(v_j \in T_2\). Each such pair is called a seed. Then we use this set of seeds to impose additional constrains on the matching to be found by some matching algorithm from the LCA-preserving matching family and use them to achieve a speed-up which is proportional to the amount of provided seeds and dependent upon their distribution over the trees. To achieve this speed-up, the condition that the provided seeds must obey is that of a planar layout, i.e. two seeds cannot cross. This condition is natural in ordered trees, but not in unordered ones since the seeds’ layout is dependent on the trees’ layout. Therefore, there might be some layout of the trees where the seeds will not cross. In this case a preprocessing procedure that finds such a layout (if one exists) is required. This “untangling” algorithm is described in due course.

The rest of the chapter is arranged as follows: the basic formalism is given in Section 3.1. Section 3.2 describes a general framework and the corresponding analysis for seeded tree align-
ment. In Section 3.3, an algorithm is presented which computes a planar layout for two unordered seeded trees, if such exists. Finally, Section 3.4 describes an application for seeded tree alignment.

The results reported in this chapter appear in [72].

### 3.1 Preliminaries and Definitions

Consider the constrained form of the tree matching problem in which the mapping of a subset of the nodes in one tree to a corresponding subset of the nodes in the other tree is given in advance. The initial node mapping is called the seed set of the matching.

**Definition 7 (mapping, seed set)** This is an extension of Definition 1. A mapping $M$ of a tree $T_1 = (V_1, E_1)$ to a tree $T_2 = (V_2, E_2)$ is a bijection $M \subseteq V_1 \times V_2$ such that for all $(v_1, v_2), (w_1, w_2) \in M$, it holds that $v_1$ is an ancestor of $w_1$ in $T_1$ if and only if $v_2$ is an ancestor of $w_2$ in $T_2$. A seed set $S$ is a bijection $S \subseteq M \subseteq V_1 \times V_2$ such that $S$ itself also is a mapping of $T_1$ to $T_2$.

Among all possible mappings, in this chapter we deal with the commonly used least-common-ancestor (LCA) preserving ones.

**Definition 8 (LCA-preserving mapping)** Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be trees, and let $M \subseteq V_1 \times V_2$ be a mapping of $T_1$ to $T_2$. $M$ is LCA-preserving if the following condition holds: if $(x_1, x_2) \in M$ and $(y_1, y_2) \in M$ then $(\text{lca}(x_1, y_1), \text{lca}(x_2, y_2)) \in M$.

We next define a new tree alignment optimization problem over pairs of seeded trees, to be applied as a constrained form of a general, pre-selected tree alignment algorithm. Therefore, let $\text{TAA}(T_1, T_2)$ denote a “black box” tree alignment algorithm, which applies a pre-selected tree alignment algorithm to an input consisting of two labeled trees $T_1$ and $T_2$. The class of tree alignment algorithms to which the seed constraint can actually be applied is discussed in the following section.

**Definition 9 (seeded tree alignment problem)** Given two trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$, a set of seeds $S \subseteq V_1 \times V_2$, and a predefined tree similarity measure $SIM$, such that $SIM(M)$ denotes a similarity score computed based on the pairs of nodes $(v_1, v_2) \in M$. The seeded tree alignment problem $\text{STA}(T_1, T_2, S, \text{TAA})$ is to find a mapping $M \subseteq V_1 \times V_2$ such that $S \in M$ and the alignment score $SIM(M)$ is maximal under this constraint.
3.2 Tree Alignment based on Seeded Nodes

In this section, we show how to efficiently apply seeded alignment on top of existing tree alignment algorithms. We note that our results apply to LCA-preserving mappings (see Def. 8). This class of algorithms includes subtree isomorphism [77, 78, 107], subtree homeomorphism [18, 87, 95], and maximum common subtree (MCS) [119] finding algorithms. For the sake of clarity of presentation, we note that all these algorithms employ a dynamic programming table where entry \([i, j]\) denotes the similarity score of subtree \(i\) of tree \(T_1\) versus subtree \(j\) of tree \(T_2\). Moreover, the time complexity of each of the above algorithms is computed by summing up the work invested in matching a subtree \(t_u \in T_1\) with a subtree \(t_v \in T_2\):

\[
O\left(\sum_{u=1}^{n} \sum_{v=1}^{m} (c(u)^x c(v)^y f(c(u), c(v)))\right)
\]

where \(|T_1| = n, |T_2| = m\), \(c(u)\) denotes the out-degree of \(t_u\), \(c(v)\) denotes the out-degree of \(t_v\), and \(f(c(u), c(v))\) is a concave function that differs from one algorithm to another along with coefficients \(x\) and \(y\). For example, unordered subtree homeomorphism can be computed in \(O(nm\sqrt{m})\) time.
using the top-down algorithm of [18] and the corresponding concave function is $\sqrt{m}$ (see Example 1 for a complete analysis).

In the discussion to follow, let the seeds contained in the initial seeds set $S$ be denoted primary seeds. Since we restrict our analysis to LCA-preserving mappings, the LCAs of the primary seeds also function as seeds, to be denoted secondary seeds (see Fig. 3.1 (left)). For the sake of simplicity of presentation we will describe the seeded tree alignment algorithm for binary trees. Extensions to non-binary trees are straightforward. Note that, given an LCA-preserving tree alignment algorithm, and given as input a planar layout tanglegram of a pair of seeded trees that are to be aligned, the corresponding seeded tree alignment could immediately be derived by extending the applied node label similarity table as follows: For each seed $s = (u, v) \in S$ such that $u \in T_1$ and $v \in T_2$, relabel the seeded nodes to $u'$ and $v'$ respectively and add two new rows and two new columns to the label similarity table – one for node $u'$ and the other one for node $v'$. Then, the similarity score for entries $[u', v']$ and $[v', u']$ is set to $\text{max}_\text{pairwise}_\text{score}$, which is the highest possible value of a node-to-node pairwise score under the predefined scoring scheme, while all the remaining entries in these two rows and columns are set to zero. This way we ensure that the above LCA-preserving tree alignment algorithms will match seeds as required.

That being said, in this section we show how to exploit the seeds to more efficiently apply the tree alignment, and avoid redundant work by restricting the computations to limited areas in the dynamic programming (DP) matrix. This “constrained-by-seeds” dynamic programming can be intuitively explained by following the example in Fig. 3.1. A regular, unconstrained application of the LCA-preserving algorithms mentioned above to the two trees in Fig. 3.1 (left) would require the computation of each and every entry in the DP table of Fig. 3.1 (right). The algorithm described below, however, will only compute the shaded rectangles along the diagonal of the table. Note that each primary seed corresponds to a single entry in the DP table whose score can be computed in an initialization step. Furthermore, each pair of consecutive seeds in $S$ (according to a planar layout) defines a rectangle in the DP matrix with a side of size at most $k$, where $k$ denotes the maximum gap between two consecutive seeds (in a planar layout), that can be filled independently of other rectangles. This is true for all entries except for the single entry in the rectangle which corresponds to a secondary seed, and whose computation depends on the availability of entries external to the rectangle. This availability, however, can be taken care of if the rectangles are processed by postorder traversal of the corresponding secondary seeds. The number of rectangles is bounded by $n/2k$, and the size of each such rectangle is bounded by $k^2$, and thus, there is an immediate $O(nk)$ bound on the number of entries that need to be computed in the table (in comparison to $O(n^2)$ entries in the unconstrained tree alignment case). Furthermore, each application of TAA is given as input two subtrees with no more than $k$ nodes.

The time complexity of seeded LCA-preserving tree alignment algorithms is formally analyzed in
Obs. 4 and demonstrated in Example 1.

We refer the reader to Fig. 3.1 (left) for the following discussion. Consider the subtree obtained during a postorder traversal of $T_1$, from node $c$ to node $d$: note that all nodes located in the left part are colored green and all nodes located in the right part are colored blue. Correspondingly, in the subtree obtained during a postorder traversal of $T_2$, from node $c'$ to node $d'$, all nodes located in the right part are colored green and all nodes located in the left part are colored blue. This correspondence of colors is explained by Obs. 3; before we state it we need the following definition.

**Definition 10** For any tree $T$ and nodes $x, y \in T$, let $t_{x,y}$ denote the subtree consisting of all nodes found in a postorder traversal of $T$, starting from node $x$ and ending in node $y$. Also, let $\text{left}_{x,y}$ and $\text{right}_{x,y}$ denote the left and the right subtrees of $t_{x,y}$, respectively.

Note that both $\text{left}_{x,y}$ and $\text{right}_{x,y}$ are rooted at $\text{lca}(x,y)$.

**Observation 3** Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be trees to be aligned and $(x_1 \in V_1, x_2 \in V_2)$ and $(y_1 \in V_1, y_2 \in V_2)$ be a pair of seeds such that $x_1 < y_1$ and $x_2 < y_2$ in the postorder traversal of $T_1$ and $T_2$, respectively. In an LCA-preserving mapping of $T_1$ to $T_2$, all nodes in $\text{left}_{x_1,y_1}$ are mapped to nodes in $\text{right}_{x_2,y_2}$. Symmetrically, all nodes in $\text{left}_{x_1,y_1}$ are mapped to nodes in $\text{right}_{x_2,y_2}$.

**Proof** By recursive invocation of Def. 8.

The seeded tree alignment algorithm starts by extending the seeds set $S$ to include the secondary seeds. Next, it orders $S$ such that all the seeds obey a planar layout, that is, there is no crossing between seeds. An algorithm to compute this layout, if such layout exists, is given in Sect. 3.3. The resulting order partitions the target trees, according to Obs. 3, into exclusive subtree-pair intervals (see Fig. 3.1 (right)). The suggested algorithm processes these subtree pairs in postorder traversal of their roots (which are paired as secondary seeds). For each such interval, it retrieves the corresponding subtrees and feeds them as input to TAA.

The pseudocode for the algorithm is given below:

A special case occurs when, upon completion, there remain unseeded nodes $X_{T_1}, X'_{T_2}$ before the first seed and $Y_{T_1}, Y'_{T_2}$ after the last seed in the trees. In order to handle these nodes, the algorithm invokes an appropriate LCA-preserving mapping once for $(X_{T_1}, X'_{T_2})$ and $(Y_{T_1}, Y'_{T_2})$ and once for $(X_{T_1}, Y'_{T_2})$ and $(Y_{T_1}, X'_{T_2})$ and picks the maximum between the two options to add to the obtained seeded matching. The resulting matching score corresponds to an entry $[r, r']$ where $r$ and $r'$ are the roots of $T_1$ and $T_2$ respectively.
Algorithm 3 Given two trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ and a set of primary seeds $S \subseteq V_1 \times V_2$, find a best possible mapping $M$ of $T_1$ to $T_2$ such that $S \subseteq M \subseteq V_1 \times V_2$.

1: procedure SEEDED MATCHING($T_1, T_2, S$)
2: $S' \leftarrow$ a set of secondary seeds based on $S$
3: $S \leftarrow S \cup S'$
4: $S \leftarrow$ layout order($S$)
5: for all primary seeds $(x, x')$ in $S$ do
6: $DP[x, x'] \leftarrow TAA(t_x, t_{x'})$
7: end for
8: $(x, x') \leftarrow$ the first secondary seed in a postorder on $T_1$
9: while $(x, x') \neq$ the last secondary seed in a postorder on $T_1$ do
10: $(y, y') \leftarrow$ the left child seed of $(x, x')$
11: $(z, z') \leftarrow$ the right child seed of $(x, x')$
12: $DP[y, y' \ldots x, x'] \leftarrow TAA(left_{x-y}, right_{x'-y'})$
13: $DP[x, x' \ldots z, z'] \leftarrow TAA(right_{y-z}, left_{y'-z'})$
14: $DP[x, x'] \leftarrow TAA(t_x, t_{x'})$
15: $(x, x') \leftarrow$ the next secondary seed in a postorder on $T_1$
16: end while
17: return $DP[r_1, r_2]$
18: end procedure

The described algorithm requires a primary seeds set $S$ to contain seeds that are leaves. However, it can be easily extended to handle non-leaf seeds by the following preprocessing procedure. For each non-leaf seed $(i, j)$ in $S$ we invoke an LCA-preserving procedure for mapping a subtree rooted at node $i \in T_1$ to a subtree rooted at node $j \in T_2$. Then the above subtrees are pruned and the seed receives a score of the matching between the two subtrees.

Lemma 9 Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be two trees to be aligned, and let $S \subseteq V_1 \times V_2$ be a primary seeds set. Given an LCA-preserving tree alignment algorithm TAA and the corresponding score function SIM, Algorithm 3 computes $STA(T_1, T_2, S, TAA)$.

Proof The condition $S \in M$ is kept by lines 5,6 of Algorithm 3. LCA-preservation is kept by the definition of secondary seeds, by adding secondary seeds to $S$, and by the fact that, in line 14 of Algorithm 3, for any seed cell $DP[i, j]$, the values all other entries in line $i$ and all other entries in line $j$ remain set to null. In lines 12, 13 of Algorithm 3, the pre-selected tree alignment algorithm TAA computes the optimal score to each one of the subtree-pairs confined by the seeds, according to the LCA-preservation constraint enforced by Obs. 3. The candidate LCA-preserving algorithms mentioned above compute the values of $DP$ in bottom-up node order and thus the postorder processing of the subtrees corresponding to secondary seeds ensures that the necessary node-values are available when needed. Therefore, by Def. 9, the resulting alignment is the best
scoring one under the seed constraints.

Restricting the computations to limited areas in the DP matrix results in a speedup of the applied, predefined tree comparison algorithms, as analyzed below.

**Lemma 10** The above framework for computing $\text{STA}(T_1, T_2, S, \text{TAA})$ yields a speedup of $\Omega((n/k)^{x+y-1} f(n, n)/f(k, k))$ over the time complexity of the corresponding, unseeded, tree alignment algorithm $\text{TAA}(T_1, T_2)$.

**Proof** Let $f(c(u), c(v))$ denote the concave function quantifying the work that a given (LCA-preserving, DP subtree-to-subtree based) tree-comparison algorithm $\text{TAA}$ applies per alignment of a subtree $t_u \in T_1$ with a subtree $t_v \in T_2$, where $c(u)$ denotes the out-degree of $t_u$ and $c(v)$ denotes the out-degree of $t_v$.

**Observation 4** $\sum_{u=1}^{k} c(u) = k$ and $\sum_{v=1}^{k} c(v) = k$.

Summing up the work over all node pairs, applying Obs. 4 to Eq. 3.1 we get:

$$O\left(\frac{n}{k} \sum_{u=1}^{k} \sum_{v=1}^{k} (c(u)^x c(v)^y f(c(u), c(v)))\right) =$$

$$= O\left(\frac{n}{k} k^x \sum_{v=1}^{n} (c(v)^y f(c(u), k)) = O\left(\frac{n}{k} k^x k^y f(k, k)\right) = O(nk^{x+y-1}(f(k, k))\right).$$

This yields a speedup of $\Omega((n/k)^{x+y-1} f(n, n)/f(k, k))$ over the time complexity obtained by applying the corresponding unseeded version of the tree comparison algorithm.

Below we give an example of one such seeded tree alignment algorithm.

**Example 1 (Top-down unordered subtree homeomorphism [18])** The algorithm for top-down unordered subtree isomorphism between trees $T_1$ and $T_2$ with $|T_1| = n_1$ and $|T_2| = n_2$ runs in $O(n_1 n_2 \sqrt{n_2})$ time, since

$$O\left(\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} (c(u)c(v) \sqrt{c(v)}) = O(n_1 n_2 \sqrt{n_2})\right)$$

When applied over a seeded tree matching, we get

$$O((n_1/k) \sum_{u=1}^{k} \sum_{v=1}^{k} (c(u)c(v) \sqrt{c(v)}) = O((n_1/k)k \sum_{u=1}^{k} (c(v) \sqrt{c(v)})) =$$

$$= O((n_1/k)k^2 \sqrt{k}) = O(n_1 k \sqrt{k})$$
Thus, if the compared trees are heavily seeded and \( k = O(1) \) then the algorithm runs in \( O(n_1) \) time and the speedup factor is \( O(n_2 \sqrt{n_2}) \).

### 3.3 Planar Tanglegram Layout

A layout of two unordered trees with additional edges forming a bijection among their leaves, is called a tanglegram [85]. These diagrams arise in host-parasite cospeciation studies and in the reconciliation of gene and species phylogenies.

**Definition 11 (Tanglegram)** A tanglegram is a triple \((T_1, T_2, S)\) where \( T_1 = (V_1, E_1) \) and \( T_2 = (V_2, E_2) \) are unordered trees, and \( S \subseteq V_1 \times V_2 \) is a seed, that is, a partial mapping of \( T_1 \) to \( T_2 \). A tanglegram is binary if both \( T_1 \) and \( T_2 \) are binary trees.

Given a tanglegram \((T_1, T_2, S)\), we will be interested in finding a way to represent the two trees in such a way that the seed does not create any crossings among the edges corresponding to seeds in that representation. We call such a representation a planar layout of the tanglegram. To define it formally, we first introduce the notion of an extension of a set (and a pair of sets) of nodes.

**Definition 12 (Extension)** Let \( T_1 \) be an unordered tree, let \( X \) be an ordered set of nodes in \( T_1 \), and let \( u \in X \) be a non-leaf of \( T_1 \). Denote by \( X' \) the ordered set \( X \) where \( u \) has been replaced by its children in some particular ordering. Then, we call \( X' \) a one-step extension of \( X \). We say that \( Z \) is an extension of \( X \) if there is a sequence of zero or more one-step extensions from \( X \) to \( Z \).

Let \( Y \) be an ordered set of nodes in an unordered tree \( T_2 \). Then, we also say that \((X',Y')\) is an extension of \((X,Y)\) if \( X' \) is an extension of \( X \) and \( Y' \) is an extension of \( Y \).

We are interested in extending the pair formed by the roots of the two trees of a tanglegram until there is no point in extending it further. The extensions are performed until no seed with seeded descendants can be found (for instance, seeded leaves satisfy this condition). In the following, we will call these nodes terminals.

**Definition 13 (Planar layout)** Let \( T_1 \) and \( T_2 \) be unordered trees with roots \( r_1 \) and \( r_2 \), respectively. A planar layout of a tanglegram \((T_1, T_2, S)\) is a pair \((x, y)\) with \( x = (x_1, \ldots, x_n) \) and \( y = (y_1, \ldots, y_n) \), such that:

- \((x, y)\) is an extension of \(((r_1), (r_2))\),

- the nodes in \( x \) and \( y \) are terminals, and
• \((x_i, y_i) \in S\) for every \(i\) with \(1 \leq i \leq n\).

**Example 2** The tanglegram to the left has a planar layout, namely: \(((a, b, d, c), (a, b, d, c))\), while the one to the right does not.

![Tanglegram Diagram]

We next describe an algorithm for finding a planar layout of a binary tanglegram. The procedure \textit{Untangle} computes the layout of a binary tanglegram by successive refinements of two lists, the ordered sets \(X\) and \(Y\), which initially contain the roots of the trees. At each iteration of the loop, a node of one of the lists is “refined,” which means that a correct ordering of its children is found and fixed for the rest of the algorithm. The loop stops when all the elements of the lists \(X\) and \(Y\) are terminal nodes of the trees; at this point, the planar layout (if it exists) is completed.

Before starting the main loop, the procedure \textit{Paths} computes a table \(P\) of Boolean values which can be understood as an extension of the bijection \(S\) to all the nodes of the trees. In particular, for any node \(u\) in \(T_1\) and any node \(v\) in \(T_2\), \(P[u, v]\) is true if and only if the subtree of \(T_1\) rooted at \(u\) has a descendant \(u'\), the subtree of \(T_2\) rooted at \(v\) has a descendant \(v'\), and \((u', v') \in S\). The pseudocode of the procedure \textit{Paths} is given below:

The computation made by \textit{Paths} can follow a dynamic programming approach. In the first place, the entries corresponding to the leaves are given by \(s\). Then, it proceeds by computing the entries \((u, v)\) where \(u\) and \(v\) are inner nodes in the trees. The value of \(P[u, v]\) is set to \textit{true} if and only if an entry \(P[u_i, v_j]\) has the value \textit{true} for some child \(u_i\) of \(u\) and some child \(v_j\) of \(v\). The cost of computing all the entries is, therefore, \(O(n^2)\).

Now we return to the main procedure.

In the refinement step, a node \(u\) in the graph \((X \cup Y, E)\) is substituted by its children \(u_1, u_2\) in such a way that no edge crossing is introduced.

The above procedure selects an ordering of the nodes \(U = \{u_1, u_2\}\) such that, replacing \(u\) by \(U\) in \(A\), the graph \((A \cup B, E)\) does not create any edge crossings. Formally, we say that \((A \cup B, E)\) has an \textit{edge crossing} if there are two nodes \(a_1, a_2\) in \(A\) and two more nodes \(b_1, b_2\) in \(B\), appearing in this order in \(A\) and \(B\), such that \(E\) contains the edges \((a_1, b_2)\) and \((a_2, b_1)\). Assuming \((A \cup B, E)\)
Algorithm 4 Given a bijection $S$ among a subset of the nodes of two trees $T_1$ and $T_2$, return the extension $P$ of $S$ to all the nodes of $T_1$ and $T_2$.

1: procedure Paths($T_1, T_2, S$)
2: for all nodes $v$ of $T_1$ in postorder do
3:   for all nodes $w$ of $T_2$ in postorder do
4:     if $v$ is a leaf and $w$ is a leaf then
5:       $P[v, w] \leftarrow \{v, w\} \in S$
6:     end if
7:     if $v$ is a leaf and $w$ is not a leaf then
8:       \{w_1, w_2\} $\leftarrow$ children of $w$
9:       $P[v, w] \leftarrow P[v, w_1] \lor P[v, w_2] \lor \{v, w\} \in S$
10:   end if
11: if $v$ is not a leaf and $w$ is a leaf then
12:   \{v_1, v_2\} $\leftarrow$ children of $v$
13:   $P[v, w] \leftarrow P[v_1, w] \lor P[v_2, w] \lor \{v, w\} \in S$
14: end if
15: if $v$ is not a leaf and $w$ is not a leaf then
16:   \{v_1, v_2\} $\leftarrow$ children of $v$
17:   \{w_1, w_2\} $\leftarrow$ children of $w$
18:   $P[v, w] \leftarrow P[v_1, w_1] \lor P[v_1, w_2] \lor P[v_2, w_1] \lor P[v_2, w_2] \lor \{v, w\} \in S$
19: end if
20: end for
21: end for
22: return $P$
23: end procedure

Algorithm 5 Given a tanglegram $(T_1, T_2, S)$, obtain a planar layout $(X, Y)$ for $(T_1, T_2, S)$. Let $r_1, r_2$ be the roots of $T_1, T_2$, respectively.

procedure Untangle($T_1, T_2, S$)

$X, Y \leftarrow (r_1), (r_2)$
$E \leftarrow \{(r_1, r_2)\}$
$P \leftarrow \text{Paths}(T_1, T_2, S)$

while $X \cup Y$ contain some non-terminal node do

$u \leftarrow$ a non-terminal node of highest degree in $(X \cup Y, E)$

if $u$ is in $X$ then

Refine($u, X, Y, E, P$)

else

Refine($u, Y, X, E, P$)

end if

end while

return $(X, Y)$

end procedure
Algorithm 6 Given a partial planar layout \((A \cup B, E)\) and a node \(u\), refine the planar layout by substituting \(u\) by its children and return \(A\) and \(E\) modified according to the refinement.

```plaintext
procedure Refine\((u, A, B, E, P)\)
    \(u_1, u_2 \leftarrow\) children of \(u\)
    for every node \(v \in B\) such that \(\{u, v\} \in E\) do
        if \(P[u_1, v]\) then
            add edge \(\{u_1, v\}\) to \(E\)
        end if
        if \(P[u_2, v]\) then
            add edge \(\{u_2, v\}\) to \(E\)
        end if
    delete \(\{u, v\}\) from \(E\)
    end for
    if \(u_1\) is an isolated node in \((\{u_1\} \cup B, E)\) then
        replace \(u\) by \(u_2\) in \(A\)
    else if \(u_2\) is an isolated node in \((\{u_2\} \cup B, E)\) then
        replace \(u\) by \(u_1\) in \(A\)
    else if not Crossings\((u_1, u_2, B, E)\) then
        replace \(u\) by the ordered set \((u_1, u_2)\) in \(A\)
    else if not Crossings\((u_2, u_1, B, E)\) then
        replace \(u\) by the ordered set \((u_2, u_1)\) in \(A\) and flip clade \(u\)
    else
        reject
    end if
end procedure
```
does not already have any edge crossings before replacing \( u \) by \( U \) in \( A \), this property is checked in the procedure \textit{Crossings} with cost \( O(n) \) by just checking if any edge adjacent with node \( u_2 \) crosses the last (in the order given by \( B \)) edge adjacent with node \( u_1 \).

Note that the whole algorithm can be thought of as the computation of an extension of \(((r_1),(r_2))\) (where \( r_1 \) and \( r_2 \) are the roots of the initial trees), which becomes a planar layout at the end. Figures 3.2 and 3.3 describe the untangling algorithm for two tanglegrams form Example 2.

![Tanglegram Diagram](image)

Figure 3.2: The refinements made by the planar tanglegram layout algorithm on the tanglegram of Example 2 (right) follow one of these two paths until a graph is found that has (under the tanglegram layout constraints) no planar layout.

In order to prove the correctness of \textit{Untangle}, we introduce the following concept.

**Definition 14 (Promising partial layout)** Let \( T = (T_1,T_2,S) \) be a tanglegram, let \( X \) be an ordered set of nodes in \( T_1 \), and let \( Y \) be an ordered set of nodes in \( T_2 \). Then, we say that \((X,Y)\) is promising for \( T \) if it extends to a planar layout of \( T \).

In the following, \( T = (T_1,T_2,S) \) will denote the binary tanglegram which is given as input to \textit{Untangle}. Additionally, \( X \) and \( Y \) will represent, as above, two ordered sets of nodes of the trees \( T_1 \) and \( T_2 \), respectively, and \( E \) will be the set of pairs of nodes kept by the algorithm. The following lemma provides an invariant for the loop in \textit{Untangle}.

**Lemma 11** If \((X,Y)\) is promising for \( T \) at the beginning of an iteration of the while loop in \textit{Untangle}, then it is promising for \( T \) at the end.

**Proof** Suppose that \((X,Y)\) is promising at the beginning of the loop, and let \( u \) be a nonterminal of highest degree. Without loss of generality, we can suppose that \( u \in X \). Note that \( u \) must have degree at least one in the graph \( G = (X \cup Y, E) \), since a promising pair cannot have isolated
nodes. Let $u_1$ and $u_2$ be the children of $u$ in the corresponding tree. Now, in case that $u_1$ ($u_2$) is isolated in $G$, the algorithm just deletes it and replaces $u$ by $u_2$ ($u_1$). Since an isolated node cannot contribute to a planar layout, the new pair $(X, Y)$ must be promising, and we are done.

Suppose now that none of $u_1$ or $u_2$ is isolated. Let $X_1$ be $X$ with $u$ replaced by $(u_1, u_2)$;
symmetrically, let $X_2$ be $X$ with $u$ replaced by $(u_2, u_1)$. We now differentiate between degree one and degree larger than one for $u$:

**Degree $= 1$.** Since $u$ has highest degree, its neighbor $v$ must have degree 1 too and, then, \{u, v\} is an edge in $G$ that is not incident upon any other edge, so it cannot be crossed over by any other edge in the one-step extensions of $(X, Y)$.

The fact that $(X, Y)$ is promising means that it must extend to a planar layout, say $(x, y)$, which is obtained either from the pair $(X_1, Y)$ or from $(X_2, Y)$. In the first case, $(X_1, Y)$ must be promising. In the second case, $(X_2, Y)$ is promising but since the subgraph of $G$ induced by \{u_1, u_2, v\} is a connected component of $G$, exchanging $u_1$ and $u_2$ and reversing children’s order in any later refinement of them leads to a planar layout. Therefore, $(X_1, Y)$ must be promising, too.

**Degree $> 1$.** Observe that we have now the following interesting situation:

**Claim 1** If $(X_1 \cup Y, E)$ does not have any edge crossing, then $(X_2 \cup Y, E)$ has some edge crossing.

The reason for the above claim is the following. Let $N(u_1)$ be the set of neighbors of $u_1$, and $N(u_2)$ the set of neighbors of $u_2$. Then, there must be at least a node in the symmetric difference of $N(u_1)$ and $N(u_2)$; otherwise, if $|N(u_1)| = |N(u_2)| = 1$, $u$ would have degree one (which is not the present case), and if $|N(u_1)| = |N(u_2)| > 1$, there would be an edge crossing. Suppose that there is a node $w$ in the symmetric difference of $N(u_1)$ and $N(u_2)$ that actually belongs to $N(u_1)$ (the other case being similar). Then, since we are assuming that $(X_1 \cup Y, E)$ has no edge crossings, $w$ must appear in $Y$ before all the neighbors of $u_2$. Then, if $Y = (y_1, \ldots, y_m)$, we have the edge \{u_1, w\} in $E$ with $w = y_i$, for some $i \leq m$, and some other edge (from $N(u_2)$) \{u_2, y_j\} for $j > i$. Now, if we exchange the order of $u_1$ and $u_2$, we get an edge crossing, as we wanted to show. (See Fig. 3.4.)
It is still left to show that the pair \((X', Y)\) is promising, for \(X'\) being the new value of \(X\) at the end of the iteration (that is, \(X' = X_1\) or \(X' = X_2\)). But now, it is straightforward since we are supposing that \((X, Y)\) is promising. If no edge crossings are found with the ordering \((u_1, u_2)\) then, by the Claim, the graph \((X_2 \cup Y, E)\) has some edge crossing and, therefore, \((X_2, Y)\) cannot be promising. But then, \((X_1, Y)\) must be promising. On the contrary, if some edge crossing is found for \((u_1, u_2)\), then \((X_2 \cup Y, E)\) must be promising since \((X, Y)\) is. In any case, the new \((X', Y)\) must be promising.

**Remark 1** Note that the choice for \(u\) made in Algorithm 2 (as a non-terminal node of highest degree) is used in the Degree 1 case of the previous proof. The only possibility that must be avoided, however, is choosing a node \(u\) of degree 1 adjacent to a node of degree \(>1\), since it would not be clear – at this stage – what the right ordering of the children of \(u\) is. So, the “highest degree” condition ensures that if \(u\) has degree 1, its (only) neighbor must have degree 1 too and then, both possible orderings of the children of \(u\) give rise to a promising pair, as it is argued in the proof of Lemma 11.

**Theorem 5** The procedure \(\text{Untangle}(T_1, T_2, S)\) computes a planar layout for \((T_1, T_2, S)\) if there is one.

**Proof** Suppose there is a planar layout for \(T = (T_1, T_2, S)\). Then, if \(r_1\) is the root of \(T_1\) and \(r_2\) is the root of \(T_2\), it is clear that \(((r_1), (r_2))\) must be promising for \(T\). By Lemma 11, the pair \((X, Y)\) is kept as a promising pair until the main loop is exited. At this point, \(X\) and \(Y\) only contain terminals, \((X, Y)\) is certainly an extension of \(((r_1), (r_2))\), and there are no crossings. Therefore, \((X, Y)\) is a planar layout.

**Lemma 12** Algorithm 5 runs in \(O(n^2)\) time and space.

**Proof** Let \(T_1\) and \(T_2\) be unordered trees with \(|T_1| = n_1\) and \(|T_2| = n_2\), and let \(n = n_1 + n_2\). The cost of Algorithm 5 is dominated by the computation of the path matrix \(P\), which takes \(O(n^2)\) time and uses \(O(n^2)\) additional space. Once \(P\) is available, the \(\text{Refine}\) procedure is called exactly once for each non-terminal node of the trees, and in each call the neighbors of the node in the graph \((A \cup B, E)\) are updated in \(O(\max(n_1, n_2)) = O(n)\) time; the \(\text{Crossing}\) procedure also takes \(O(n)\) time. Therefore, the \(\text{Untangle}\) procedure runs in \(O(n^2)\) time.

### 3.4 Applications

Matching and aligning trees is a recurrent problem in computational biology. Two prominent applications are the comparison of phylogenetic trees [20, 24, 81, 86, 108, 110, 121] and the
comparison of RNA structures [53, 69, 109, 128]. In what follows, we describe an example motivated by evolutionary studies of RNase P RNAs and their target tRNAs; it is interesting as it demonstrates the need for seeded tree alignments for both ordered and unordered trees.

Figure 3.5: The known secondary structures for two RNase P sequences and the corresponding coarse-grain trees. (left) \textit{E. coli} RNase P, based on [44], shaded its represent conserved loci. (right) \textit{M. barkery} RNase P obtained from the RNase P database http://www.mbio.ncsu.edu/RNaseP/.

Figure 3.6: Seeded tree alignment for the \textit{E. coli} versus \textit{M. barkery} RNase P secondary structures shown in Fig. 3.5. Dark vertices represent conserved loci, dotted lines represent alignment seeds.

Ribonuclease P is the endoribonuclease responsible for the 5’ maturation of tRNA precursors [44]. RNase P is a ribonucleoprotein in all organisms, but is best understood in Bacteria, in which the RNA component of the enzyme is by itself catalytically proficient \textit{in vitro} (it is a ribozyme). The structure of bacterial RNase P RNA has been studied in detail, primarily using comparative methods [84, 52, 38, 126]. Bacterial RNase P RNAs share a common core, and synthetic minimal RNase P RNAs consisting only of these core sequences and structures are
catalytically proficient. Structural variation in RNase P RNA is predominated by variation in the presence or absence of helical elements and in variation of the size of the distal regions of these helices. However, there is additional variation in the form of small differences in the lengths of helices, loops and joining regions. In terms of RNA secondary structure tree alignment, this means that the operations applied in transforming one tree to another consist of subtree deletions and insertions as well as homeomorphic node insertions and deletions in ordered rooted trees (see Fig. 3.5).

Recently, sequences encoding RNase P RNAs of various genomes have been determined (see the RNase P database, http://www.mbio.ncsu.edu/RNaseP/). This broad sampling of RNase P RNAs allows some phylogenetic refinement of the secondary structure, and reveals patterns in the evolutionary variation of sequences and secondary structures. In [44], the extent and patterns of evolutionary variation in RNase P RNA sequence and structure were studied, in both bacterial and archaeal species, and it was shown that highly-conserved bases are scattered throughout the sequence and secondary structure, and are concentrated in the vicinity of the pre-tRNA binding surface of the tertiary structure. Furthermore, there are several helices, both in the core and periphery of the RNA, that are conserved in sequence at the base and terminal loop but are extremely variable in sequence along the length of the helix. The proximal ends of these helices are located within important conserved sequence and structure, and interact at their terminal loops in secondary or tertiary contacts elsewhere in the molecule. A detailed description of the conserved loci is given in [44] and shown in Fig. 3.5. In terms of RNA secondary structure tree comparison, this means that in a biologically correct alignment of two RNase P trees, the nodes corresponding to the conserved loci should be mapped to each other (“alignment seeds”), as shown in Fig. 3.6.

The need to align seeded tree-pairs also arises in applications where the bioinformatics data is represented in the form of unordered trees. To demonstrate this, consider the example in Fig. 3.7, which illustrates the reconciliation of a phylogenetic tree based on archaeal RNase P structures with the phylogenetic tree based on archaeal rRNA structures. This figure is based on a study by Harris et al. [46], where a detailed comparative analysis of archaeal RNase P RNA structures is reported, based on 37 sequences from a wide range of archaeal species. The RNase P RNA sequences were rigorously aligned in a comparative analysis of secondary structure, providing an opportunity to compare phylogenetic relations derived from RNase P RNA sequences with those derived from small subunit ribosomal RNA sequences from the same group of organisms [75].

Although the RNase P RNA sequences generally recreate trees similar to those based on rRNA, a significant exception is the placement of the sequence from Archaeoglobus fulgidus. In rRNA-based trees, this genus lies on a branch distinct from the other major euryarchaeal groups, separating from the other groups at approximately the bifurcation between methanobacteria and
Figure 3.7: Seeded phylogenetic unordered tree alignment, in the context of horizontal gene transfer prediction. (left) The tanglegram formed by connecting, via seed edges, the phylogenetic tree based on archael RNase P structures [46] with another phylogenetic tree based on archael RNA [50]. The seed edges for the two direct neighbors, according the RNase P RNA tree: Archeaglogi and Methanococci, which are putatively involved in RNase P RNA horizontal transfer [46], were omitted. (right) The planar layout of the tanglegram.

The above analysis could be formulated as a seeded unordered tree alignment, as follows (see Fig. 3.7). Connect each leaf from the RNase P RNA tree with the corresponding (same species) leaf from the ssu-rRNA tree, if such exists. Note that the layout of two unordered trees with additional edges forming a bijection among their leaves is called a tanglegram [85]. It is easy to see that the seeded unordered trees can be aligned if the input trees can be put in a non-crossing representation (in other words: if the tanglegram formed by the input trees together with the seed has a planar layout). Correspondingly, when formulating the problem raised by [46] as that of seeded unordered tree alignment: if the tanglegram formed by the two seeded RNA trees has a planar layout, and the two trees agree, then there is no basis for a lateral transfer hypothesis. In the above example, however, the tanglegram formed by the two RNA trees can be untangled, and the two trees can be aligned after removing the seed edges corresponding to the two new neighbors (by RNase P RNA homology) Archeaglobi and Methanococci. This supports the hypothesis of a lateral transfer of the gene encoding RNase P RNA from Archeaglobi to Methanococci, or vice versa.
3.5 Discussion

This chapter described a novel approach to utilizing prior knowledge about the desired matching in order to provide a result consistent with the expectation and to improve the time complexity of an entire family of algorithms. This is achieved by filling in only certain parts of a dynamic programming table and the invocation of an existing matching algorithm as a black-box on relatively small subtrees. This approach may be very useful in such bioinformatics applications as comparison of RNA secondary structures and aligning of phylogenetic trees. One interesting problem that arises when dealing with unordered tree matching is the requirement of a planar layout of the seeds set. This problem is solved by the untangling algorithm also presented herein.

The extension of the Untangle procedure (Algorithm 5) to compute the planar layout of an arbitrary – not necessarily binary – tanglegram is an interesting open problem. While the top-down approach of the binary case is maintained, the refinement of the nodes — in the general case — implies replacing a node by an arbitrary number of new nodes. In order to sort them correctly, so that the whole graph is kept planar, a characterization in terms of caterpillars [35] may be used, together with a technique for grouping the nodes which cannot be correctly ordered at some particular step. The optimization problem of finding the smallest number of seeds to be removed from a tanglegram in order to obtain a planar layout is another interesting line of future research.
Chapter 4

Alignment of Metabolic Pathways

Genome-scale metabolic networks are now being reconstructed for a variety of organisms such as *Escherichia coli*, *Saccharomyces cerevisiae*, and human. The wealth of information regarding the chemical reactions that take place within a cell and the corresponding enzymes that catalyze these reactions is currently stored in several public databases, including KEGG [56], EcoCyc [62] and SGD [17]. These databases maintain information about complex cellular processes, such as metabolism, signal transduction and gene regulation, by storing the corresponding networks of interacting molecules in digital form, often as graph-based pathway diagrams. The majority of these databases provide tools (e.g. [83]) for pathway visualization and for queries on pathway components such as substrates, products and reactions. However, the need arises for good tools capable of e.g. searching for homologues to a query pathway in a collection of known pathways, and of aligning two pathways to locate conserved pathway fragments.

Pathway alignments should reflect both the similarity (rather than identity) between the enzymes that participate in the aligned pathways, as well as between their topologies. The need for advanced tools for pathway analysis will increase over the next several years as biologists begin not only to inspect existing pathways but also to redirect and re-engineer metabolic pathways. The latter objective, called *Metabolic Pathway Tinkering* [80], requires thorough analysis of metabolic pathways, and brings up the need for formalizing specific, flexible queries on pathway databases.

Work to date on pathway searching has been limited to heuristics that try to capture certain properties of the underlying graphs and use them as measures of similarity, as in [82, 125], and to visual inspection, sometimes aided by tools such as described in [102]. Another attempt was undertaken by Tohsato *et al.* [118] who proposed a method for multiple alignment of metabolic pathways, but restricted the pathways’ topology to chains (or strands). Kelley *et al.* [59] presented an algorithm that given a linear pathway as a query randomly decomposes the text graph into linear pathways to find a homologous pathway among these. Also, for linear pathways Shlomi *et
al. [112] proposed an algorithm that is based on random coloring. Related work, which data-mines chains in protein-protein networks, is described in [60]. Recently, Koyutürk et al. [66] presented a related mining approach where frequently occurring patterns (that can be general graphs) are detected in biological networks. Still, they do not address the search scenario, and — moreover — they state that the issue of approximate (rather than exact) matching is an important open problem. A somewhat different approach was presented in [67] where a query consists of a set of labels without explicitly provided a topology and the algorithm searches for a subgraph with maximum nodes’ similarity to a query set. This type of queries, however, is out of the scope of our study.

In order to comprehensively search and mine metabolic pathways we developed MetaPathway-Hunter, a novel tool for pathway alignment which is based on the Approximate Labeled Subtree Homeomorphism approach presented in Chapter 2. Our alignment model, the algorithm supporting it, and its implementation are described in Section 4.1. We employed MetaPathwayHunter to conduct a study on the similarities and variations in the metabolic networks of two organisms, E. coli and S. cerevisiae, that serve as model organisms for pro- and eukaryotes, respectively, and observed several biologically interesting findings. Furthermore, we provide a description of meta-pathway queries that enable the user to probe the metabolic pathways database in a most flexible yet powerful manner. The experiments, their results, and the usage of meta-pathway queries are described in Section 4.3. We conclude with a brief discussion and some suggestions for future work. Also, the MetaPathwayHunter user’s manual is provided in Appendix A.

The results reported in this chapter appear in [88].

4.1 Modeling and Computational Approach

In order to compare pathways to each other using a quantitative measure, we must represent them as mathematical objects that lend themselves to effective computation. Here we represent a pathway by an enzyme graph whose nodes correspond to enzymes that catalyze the pathway’s reactions, and the edges connect two nodes if for the corresponding enzymes the product of one serves as the substrate of the other. When computing the similarity between metabolic pathways, we take into account both the resemblance between any two corresponding nodes in the pathway graph as well as the likeness between the pathways’ network structure. The former reflects the similarity between matched enzymes, based on functional homology, and the latter checks for topological similarity between the graphs in a biologically meaningful way.

When comparing two pathways we try to align them to each other as best we can. Similarly to the alignment of genomic and proteomic sequences, we match pathways up in such a way that similar ingredients are paired with each other while minimizing the differences between them.
These differences pertain both to the nodes, where enzymes of similar function are deemed close to each other, as well as to the connections between the nodes, namely the edges and paths that form the structure of the pathway.

As in sequence alignment, the closeness between two pathways is reflected by a score that is obtained by computing a function that measures the distance in a meaningful manner. Our method exhaustively computes all optimal solutions under a given scoring model. Furthermore, suboptimal solutions (up to a predefined threshold score) are reported, ranked by their statistical significance. This is clearly preferable both to naive visual inspection which is expensive and prone to human errors, as well as to standard heuristic search methods which are likely to overlook some of the relevant results.

In this section we first describe our graph similarity measure and define the alignment score. They are limited to tree-like graphs in order to allow efficient alignment based on graph matching algorithms which are described next. Then we show how this method is highly applicable to metabolic pathways and explain how we compute the statistical significance of the score. We conclude this section with a few details concerning the implementation of our tool.

As mentioned previously, metabolic pathways can be modeled by enzyme graphs. Given this model, the alignment between two given pathways can be obtained by finding subgraph iso- or homeo-morphism between the corresponding graphs which is known to be NP-complete. However, our study shows that the topology of most metabolic pathways can be easily cast as multi-source trees or transformed to them without much loss of generality, as cycles are quite rare in this data. A multi-source tree is a directed acyclic graph (DAG), whose underlying undirected graph is a tree (see Figure 4.2), where some of the nodes can have several incoming as well as several outgoing edges.

We base our metabolic pathway alignment engine on the approximate labeled subtree homeomorphism (ALSH) model presented in Chapter 2. The reasons for that that are both biologically and computationally driven. Biologically, a single enzyme in one pathway may replace a few consecutively acting enzymes in another pathway. The replacement can take place if the replacing enzyme is multifunctional and can thus catalyze several consecutive reactions, or if the enzyme uses an alternative catalysis that leads directly from the initial substrate to the final product. Note that enzymes that catalyze just a single reaction are more likely to be replaced than those that catalyze more reactions, for both biochemical and parsimony-related reasons. Translating this biological description into graph terms implies that degree-2 nodes may be deleted from the graph, a behavior which is perfectly captured by subtree homeomorphism.

Computationally, the advantage of subtree homeomorphism over the more complex models (such as [64]) is in that it has tractable solutions. Complicating the model by e.g. allowing the deletions from both sides would render the problem intractable.
The model we employ extends previously known exact tree matching models which allowed nodes to be matched only if their labels were identical. Our model, on the other hand, is based on an inexact pattern matching algorithm i.e. it enables matching two nodes with distinct labels, and scores the match according to the similarity between the nodes.

We observe that the ALSH problem on directed multi-source trees is a sparse instance of ALSH on unrooted unordered trees (the fact that the edges are directed reduces the number of possible mappings). Thus, an algorithm for directed multi-source trees can be obtained by extending the Approximate Subtree Homeomorphism algorithm without increasing the algorithm’s complexity.

Let \( T^r = (V_T, E_T, r) \) be the text tree which is rooted in \( r \), and \( P^{r'} = (V_{P'}, E_{P'}, r') \) be the pattern tree which is rooted in \( r' \), respectively. Let \( p^r_u \) denote a subtree of \( P^{r'} \) which is rooted in node \( u \) of \( P^{r'} \), and \( t^r_v \) denote a subtree of \( T^r \) which is rooted in node \( v \) of \( T^r \). Let \( y_1, \ldots, y_c(v) \) be the children of \( v \in T^r \), and let \( x_1, \ldots, x_{c(u)} \) be the children of \( u \in P^{r'} \). (Note that \( c(u) \leq c(v) \), as no deletions are allowed from the pattern.) We define \( \text{AlignmentScores}[u \in V_p, v \in V_T] \) as follows: For each node \( v \in V_T \) and for each node \( u \in V_{P'} \), \( \text{AlignmentScores}[u, v] \) is the maximal LSH similarity score between any subtree \( p^r_u \) of \( P^{r'} \) and a corresponding homeomorphic subtree \( t^r_v \) of \( T^r \), if such exists. Otherwise, \( \text{AlignmentScores}[u, v] = -\infty \).

\( \text{AlignmentScores}[u, v] \) is computed, using procedure \( \text{ComputeScoresForTextNode}(v, u) \) from Algorithm 1 in Chapter 2.

Figure 4.1 exemplifies a single call to procedure \( \text{ComputeScoresForTextNode}(u, v) \) for aligning a pair of subtrees.

Upon the call to procedure \( \text{ComputeScoresForTextNode}(u, v) \), the scores for comparing each subtree of \( u \) (rooted at either \( x_1 \) or \( x_2 \)) with each subtree of \( v \) (rooted at \( y_1, y_2 \) or \( y_3 \)) have already
been computed and stored in the corresponding cells of the dynamic programming table \( DP \). Note that the score for comparing \( x_1 \) with \( y_3 \) is 1, obtained as the sum of two terms: a score of 2 for aligning an “a” with an “A” plus a penalty of -1 for deleting a “D” (see the scoring table \( \Delta \) in Figure 4.2). Similarly, the score for aligning \( x_2 \) with \( y_3 \) is 1, which is the score of aligning a “b” with a “D”. (By definition of subtree homeomorphism, there is no penalty for deleting the subtree of \( y_3 \) that is labelled with an “A”). In the same manner, the best score for aligning of \( x_1 \) with \( y_2 \) is -3, obtained by matching an “a” directly to a “C” and deleting the subtrees of \( y_2 \) which are labeled with an “F”. Similarly, the score for aligning \( x_2 \) with \( y_2 \) is -2. All other subtree pairs are composed of two leaf nodes and therefore their score has been previously set by direct lookup in the scoring table \( \Delta \). Procedure \( \text{ComputeScoresForTextNode}(u,v) \) constructs the bipartite graph \( G \) shown in Figure 4.1 with bipartition \( X \) and \( Y \), where \( X = \{x_1, x_2\} \) is the set of children of \( u \), \( Y = \{y_1, y_2, y_3\} \) is the set of children of \( v \), and each node in \( X \) is connected to each node in \( Y \). The weight of an edge connecting vertices \( x_i \) and \( y_j \) is set to the previously computed value \( DP[x_i, y_j] \) which is the score for the alignment of the subtree rooted at \( x_i \) with the subtree rooted at \( y_j \).

The value \( DP[u, v] \) is then computed as the maximum between the following two terms:

1. The node-to-node similarity value \( \Delta[u, v] = +2 \), plus the assignment score for \( G \) which is also +2, obtained by matching \( x_1 \) with \( y_3 \) and \( x_2 \) with \( y_1 \). This term yields a total score of +4.
2. The score for comparing node \( u \) with the best child of \( v \) is -7 and the penalty for deleting node \( v \) is -1, so this term yields a total score of -8.

Since the term contributed by the bipartite matching yields a score which is better than the score suggested by deleting node \( v \), entry \( DP[u, v] \) will finally be set to the value of +4.

### 4.1.1 Extensions to Directed Multi-Source Trees

The Approximate Labeled Subtree Homeomorphism algorithm described above can be easily extended to support unrooted, unordered trees as follows. Let \( T = (V_T, E_T) \) and \( P = (V_P, E_P) \) be two unrooted trees. The ALSH between \( P \) and \( T \) could be computed in a naive manner as follows. Select an arbitrary node \( r \) of \( T \) to obtain the rooted tree \( T_r \). Next, for each node \( u \in P \) compute the rooted ALSH between \( P^u \) and \( T_r \). Clearly, such a strategy entails the computation of alignments of subtree pairs \( (p^u, t_r^u) \) for each \( u \in P \) and \( v \in T \). We refer the interested reader to [87] for a more sophisticated variation of this algorithm.

We next turn to handle multi-source trees. Such trees are DAGs whose underlying structure is an unrooted, unordered tree, and therefore alignments corresponding to potential mappings between subtree pairs \( (p^u, t^v) \), such that \( u \in P \) and \( v \in T \), will be considered. However, here we filter-out subtree alignments that map together edges of conflicting direction. For example,
Figure 4.2: Approximate Labeled Subtree Homeomorphism. For each node, the label is written inside the circle and the variable name assigned to the node is written externally. The node-label similarity scores are specified in Table \( \Delta \). Note that the deletion score is set to \(-1\). A subtree in the text that is homeomorphic to the pattern is circled by the dashed line. The LSH score for this alignment is 7.

Consider the potential mapping between subtrees \( t'_r \) and \( p'_u \) in Figure 4.2. The following hierarchy is defined on the neighbors of node \( u \) in \( p'_u \): node \( r' \) is denoted the "parent" of \( u \) while nodes \( x_1 \) and \( x_2 \) are denoted the "children" of \( u \). Similarly, in \( t'_v \) node \( r \) is the parent of node \( v \) and nodes \( y_1 \) and \( y_2 \) are the children of \( v \). Note that node \( u \) has two incoming edges to its children \( x_1 \) and \( x_2 \) in \( p'_u \), while in \( t'_v \) node \( v \) has one incoming edge from child \( y_2 \) and one outgoing edge to child \( y_1 \). When computing ALSH for multi-source trees, a mapping between two nodes is forbidden if the directions of the edges connecting each node to its designated parent disagree. Furthermore, by definition of subtree homeomorphism, each child of \( u \) must be mapped to a child of \( v \), and therefore the algorithm for ALSH on multi-source trees will set the alignment score for the subtree pair \( (p'_u, t'_v) \) to \(-\infty\). Thus, the additional edge-direction information in multi-source trees restricts the number of possible mappings by adding the requirement that both the number of the incoming edges of \( u \) and the number of outgoing edges of \( u \) must be smaller than or equal to the numbers of the incoming and outgoing edges of \( v \), respectively.

As for legitimate subtree mappings, the weighted bipartite matching computation is updated as follows to utilize the edge-direction information in multi-source trees: consider the bipartite graph \( G = \{ X \cup Y, E \} \), where \( X \) denotes the children of \( v \) in \( t'_v \) and \( Y \) denotes the children of \( u \) in \( p'_u \). A vertex \((x_i, y_j)\) will now be included in \( E \) if and only if the direction of the edge connecting \( x_i \) to \( u \) is similar to the direction of the edge connecting \( y_j \) to \( v \). Therefore, we get a sparse bipartite graph, which could actually be split into two separate, smaller bipartite graphs: one
corresponding to matchings of incoming-edge neighbors of \( u \) and \( v \), and the other for matching outgoing-edge neighbors.

### 4.2 Application to Metabolic Pathway Analysis

In this section we first describe our method and data sources and then analyze their significance.

**Metabolic Data Sets.** Metabolic pathways of *E. coli* were extracted from the EcoCyc [62] database and metabolic pathways of the yeast *S. cerevisiae* were extracted from SGD [17]. Both databases combine automatic pathway creation based on gene annotations as well as manual curation. Our dataset contained all pathways composed of two reactions or more that appear in these databases for these organisms (113 for *E. coli* and 151 for *S. cerevisiae*).

Note that the text graph rarely contains pathways whose underlying undirected graph is cyclic. In the seldom case of directed cycles (fewer than 10 per organism), we generated alternative multi-source trees that cover all the possible cycle-splitting variations. In the special case of DAGs which cannot be cast as multi-source trees, duplication and splitting is performed on those vertices where two incoming edges meet. This fits well with biology as the distinct paths correspond to alternative metabolic pathways.

**Alignment Scoring.** Similarly to sequence alignment, the suggested notion of pathway alignment is based on edit operations that include node substitution and node deletion (the latter relating only to the text). Alignment scoring is composed of node substitution scores that are rated by a label substitution table, and node deletion scores modelling gaps in the pattern which entail a fixed penalty. Below we describe the scoring scheme used for these two operations.

To build a label substitution table we associated each enzyme with its EC (Enzyme Commission) classification - a numbering system consisting of four sets of numbers that categorize the type of the catalyzed chemical reaction. Since an EC classification is functional, enzymes with similar EC classifications are functional homologues, but do not necessarily possess any sequence similarity. The actual values of the label substitution table were determined according to the following definition from [118]:

**Definition 4.** For an enzyme class \( h \), \( C(h) \) denotes the number of enzymes whose classes are included under \( h \). \( I(h) \), the information content of \( h \), is defined as

\[
I(h) = -\log_2 C(h)
\]

For two enzymes \( e_i \) and \( e_j \), if their lowest common upper class is \( h_{ij} \), then we consider \( I(h_{ij}) \) to express the similarity between \( e_i \) and \( e_j \).

Note that we look for the smallest common subtree that contains both enzymes. Therefore
if two enzymes are far apart in the EC classification their smallest common subtree will contain many leaves and thus their similarity level will be low. Otherwise their smallest common subtree will contain only a few leaves and their similarity level will be higher. Hence $I(h)$ increases with
the similarity.

The node deletion score (i.e. gap penalty) reflects the tradeoff between a gap and a mismatch. As the gap penalty increases, the algorithm tends to match distant enzymes to avoid gaps. Conversely, a gap penalty of zero enables alignments of evolutionary remote pathways, where only bits of the pathways are conserved, to score highly. As different values may suit different needs our tool enables users to set this parameter per execution.

**Statistical Significance of Alignments.** The statistical significance of each alignment is based on $p$-value calculation. The $p$-value of an alignment of a pathway query with score $s$ was computed by executing the same query against 100 random pathway graphs, and counting the fraction of graphs containing an alignment that received score $s$ or higher. A random pathway graph is a graph containing the same set of nodes and the same number of edges as the original graph, such that the degree of each node in the random graph is equal to its degree in the original graph. Random pathway graphs were generated from the original pathway graph by a long series of random edge switches, as described in [76].

The $p$-value cutoff used in our analysis is 0.01. We denote pathway pairs with at least one statistically significant alignment between them as *significantly aligned pathway pairs*. To assess whether the number of significantly aligned pathway pairs in the inter-species comparison and in the intra-species comparison deviate significantly from the number expected by pure chance at a cutoff of 0.01, we used the exact binomial test $(k, n, p)$ per comparison. This test computes the probability of having at least $k$ successes in $n$ Bernoulli experiments with probability $p$ for success. Here $k$ is the number of significantly aligned pathway pairs, $n$ is the total number of aligned pathway pairs, and $p$ is 0.01. This test was performed using the *R* project for Statistical Computing ([http://www.r-project.org](http://www.r-project.org)).

**Implementation Details** The algorithm was implemented as a *MetaPathwayHunter* tool using a combination of C++ code and a Java-based GUI in order to allow web applet-based usage. It runs on any Intel Pentium-based computer under the Microsoft Windows operating system (Version 2000 and higher). It does not require any special purpose hardware or other licensed software.

For each query, the tool reports the all best matches per pathway, sorted by score and statistical significance, and produces an output that graphically superimposes the query upon the aligned metabolic pathway. An example of the *MetaPathwayHunter* output can be seen in Figure A.5 in Appendix A. Also, as the GUI was recently upgraded, the diagrams which appear in the Results section differ slightly from the current graphic interface. As for determining the gap penalty,
manual inspection of the data revealed that most pathway alignments include at most one gap in a row. Considering that the worst mismatch in the EC classification is scored $-8.17$, we set the default gap penalty to $-3$, which allows for two consecutive gaps followed by a mismatch between closely-related enzymes, or for one gap and a mismatch between more distant enzymes.

The tool and the considered data sets can be downloaded from http://www.cs.technion.ac.il/~ole-gro/metapathwayhunter/. The user’s manual appear in Appendix A of this thesis.

4.3 Results

We applied our approach to the genome-scale metabolic networks of the bacterium E. coli and the yeast S. cerevisiae, as these are the two extensively studied model organisms representing the prokaryotic and eukaryotic kingdoms, respectively. We ran all-against-all alignments, namely taking each metabolic pathway as a query and aligning it against all other pathways in our dataset. We also used our tool to data-mine the pathway database with a meta-pathway query.

The runtime of the all-against-all benchmark, where query sizes ranged from 2 to 41 nodes, was measured. The entire process, including the I/O overhead of reading the pathways and recording the alignment information for successful matches, took 3.66 hours to complete on a regular desktop machine (Pentium 4, 2.6GHz clock, 512MB RAM). This yields an average of 47 seconds per query.

Below we describe results relating to both inter- and intra-species alignments, and conclude by demonstrating the power of metapathway queries in biologically relevant scenarios.

4.3.1 Inter-species Alignments

We performed all possible alignments between the 113 E. coli pathways and the 151 S. cerevisiae pathways. This analysis resulted in 610 pathway pairs that had at least one statistically significant alignment between them ($p \leq 0.01$). This number was statistically significantly greater than the randomly expected fraction of 1% ($p < 2.2e^{-16}$ using the exact binomial test). The significant alignments span most types of metabolic pathways, such as amino acid biosynthesis and fatty acid degradation, as 63% of the E. coli pathways and 66% of the S. cerevisiae pathways had at least one statistically significantly aligned pair-mate from the other species. In order to evaluate more carefully the degree of conservation between the metabolic networks of the two species we examined the alignments of the analogous metabolic pathways in E. coli and S. cerevisiae. Out of the 80 analogous pathways 62 pathways were found to be statistically significant ($p \leq 0.01$). This implies that, despite the evolutionary distance between E. coli and S. cerevisiae, a considerable fraction of their metabolic networks is conserved.
The conservation between species is not limited to small pathways, as demonstrated by the alignment of the analogous metabolic pathways of phenyl-alanine, tyrosine, and tryptophane biosynthesis in *E. coli* and *S. cerevisiae* (*s* = −4.28, *p* < 0.01). This pathway consists of 17 enzymes arranged in a star-like topology, turning the substrate erythrose-4 phosphate into one of the three amino acids phenyl-alanine, tyrosine, or tryptophane (see Figure 4.3.1). In spite of its size the pathway is almost identical between the two species, implying a common ancestral pathway. Indeed, it has been suggested that the major amino acid biosynthesis pathways were established before ancient organisms diverged into the three kingdoms of Archaea, Bacteria, and Eukaria [48].

The analogous pathways of phenyl-alanine, tyrosine, and tryptophane biosynthesis in *E. coli* and *S. cerevisiae* provide a stimulating example for the power of our tool in discovering interesting biological phenomena. Inspection of their alignment reveals that the two pathways are identical except for a single mismatch within an intermediate enzyme in the biosynthesis of tyrosine, carried out by TyrA in *E. coli* (labeled 1.3.1.13) and Tyr1 in *S. cerevisiae* (labeled 1.3.1.12). The two enzymes catalyze almost identical reactions however TyrA uses NAD+ as an acceptor while the *S. cerevisiae* enzyme uses NADP+ instead. Intriguingly, upon aligning their protein sequences using BLAST no significant sequence similarity was found between the two enzymes. The two enzymes appear to be true functional orthologs, resulting either from convergent evolution where non-homologous proteins converged to a similar function, or else from divergent evolution that changed the protein sequences but maintained their function. This example asserts our choice of EC classification as our scoring scheme since only by using a functional classification, in contrast to sequence based classification, could such a phenomenon be detected.

Gaps in the alignment of two pathways may hint to additional intriguing evolutionary phenomena. An example is the gap found upon comparing homoserine to methionine biosynthesis in *E. coli* vs. *S. cerevisiae* (*s* = −13.15, *p* < 0.01), depicted in Figure 4.3.2. In *S. cerevisiae* this pathway consists of a chain of three reactions catalyzed by three different enzymes. In *E. coli* the pathway consists of a chain of four reactions catalyzed by four different enzymes. The middle reaction in *S. cerevisiae*, catalyzed by Met17, is analogous to the succession of the two middle reactions in *E. coli*, catalyzed by MetB and MetC. Biologically, this implies that the functionality of Met17 in *S. cerevisiae* is comparable to the combined functionality of two enzymes MetB and MetC in *E. coli*. Moreover, all three enzymes are sequence homologues. This may hint to an interesting case of either gene fusion in *S. cerevisiae* or gene duplication in *E. coli*. Further investigation is needed to uncover the biological scenario that led to this incident; however, the finding that these enzymes participate in a common metabolic pathway provides a first step in this direction.
Figure 4.3: The top-scoring inter species alignments. Each node represents a match: the upper part represents the query enzyme, and the lower part represents the text enzyme. Color shades reflect enzyme homology. [1] The phenylalanine, tyrosine and tryptophan pathway of E. coli vs. S. cerevisiae ($s = -4.28, p < 0.01$). [2] Homoserine and methionine biosynthesis of E. coli vs. S. cerevisiae ($s = -13.15, p < 0.01$).

### 4.3.2 Intra-species Alignments

Intra-species alignments may provide researchers with the ability to trace the evolution of metabolism within a species. For example, the finding that pathways within a species resemble each other may imply that they arose during evolution due to instances of gene duplication followed by divergence. To demonstrate the abilities of our tool we executed all-against-all intra-species queries, where each pathway was aligned against all other pathways within the same species.

The all-against-all alignments in E. coli and in S. cerevisiae resulted in 187 significantly aligned pathway pairs in E. coli, and 262 such pairs in S. cerevisiae. The number of such pathways in E. coli is statistically significantly greater than the randomly expected fraction of 1% ($p < 4.101e^{-07}$ using the exact binomial test). The same computation for S. cerevisiae gave a statistical significance score of 0.01938. Statistically significant alignments were found for 66% of the pathways in E. coli and 62% of the pathways in S. cerevisiae, demonstrating a surprising amount of intra-species similarity within the metabolic networks.

The pathways of biosynthesis of the amino acids valine, leucine, and isoleucine (see Figure 4.4.1) provide an example for the power of intra-species alignments. The three amino acids belong to the class of hydrophobic amino acids. Valine and leucine are synthesized from the same substrate and share most of the pathway; isoleucine is synthesized from a different substrate. The intra-species alignments revealed that valine and isoleucine have identical biosynthesis pathways ($s = 0, p < 0.01$) in both E. coli and S. cerevisiae, and even employ the same set of enzymes. This
Figure 4.4: The top-scoring intra species alignments. [1] The isoleucine vs. valine biosynthesys pathways of *S. cerevisiae* (*s* = 0, *p* < 0.01) alignment. [2] The trehalose anabolism pathways of *S. cerevisiae* vs. sucrose biosynthesis pathway of *S. cerevisiae* (*s* = −9.58, *p* < 0.01). [3] The tyrosine biosynthesis of *E.coli* vs. the phenilalanine biosynthesis of *E. coli* (*s* = −8.23, *p* < 0.01).

substantiates the hypothesis that the biosynthesis of the three amino acids arose from a common ancestral amino-acid biosynthesis pathway [65]. Moreover, the degradation of the three amino acids, similarly to their biosynthesis, involves identical enzymes. Hence the entire metabolism of these three amino acids seems to stem from a single ancestral pathway.

### 4.3.3 MetaPathway Queries

So far we have discussed cases in which a user provides a specific metabolic pathway as a query. However in some cases a user may query the tool using only a partial skeleton of a certain pathway. The output of the pathway alignment tool may then identify the entire pathway scheme. One approach that is likely to benefit from this option is metabolic pathway tinkering [80], where metabolic pathways are redirected and re-engineered in order to supply certain products. To answer such needs and others we provide the possibility to form and pose a meta-pathway query.

A meta-pathway query is a pattern containing the essential enzymes as nodes and a suggested structure of their (not necessarily direct) interactions. Note that in our model no deletions are allowed in the pattern, hence it is important for all putative enzymes to appear in the pattern. Furthermore, our notion of homeomorphism allows us to represent indirect interactions as single edges in the pattern; the gap penalties must be adjusted when using the algorithm in this mode so as to increase the chances of finding chain reactions.

Meta-pathway queries may be of significant value in two likely scenarios. The first is when a user wishes to discover if two or more enzymes of interest are metabolically connected. This
may serve to understand the effect of a mutation in one enzyme on the performance of another, for example upon analyzing functional profiles of gene-deletion mutants [40]. A second scenario is when a user has limited knowledge of a certain pathway, and would like to uncover the entire pathway.

An example for the latter is given in Figure 4.5, where the query consisted of a hub enzyme and its adjacent enzymes (see Figure 4.5.1). The tool reported two significant alignments (see Figure 4.5.2), the \textit{E. coli} allantoin degradation pathway and the \textit{S. cerevisiae} ureide degradation pathway. Both pathways degrade the same substrate to three different products in \textit{S. cerevisiae} and to two of these three in \textit{E. coli} (note that the gap penalty was set to zero to allow for maximal degrees of freedom during the search). The ability to detect these related but not identical pathways through a common core demonstrates the power of meta-queries where knowledge of the entire pathway and its homologues is lacking.

### 4.4 Discussion

We have presented a new formulation for an emerging problem in bioinformatics, namely the need to find pathway patterns in larger metabolic pathway texts. Our formulation includes a score that combines both topological as well as naming similarities in a comprehensive manner. Moreover, this formulation gives rise to efficient algorithms [87] that are able to deal with more complicated...
network structures than have been handled to date.

We have implemented these algorithms and embodied them in a working tool that can be effectively used by life science researchers. Our new tool yields more comprehensive queries than those supported by previous tools, which were restricted to chain topology and therefore could not capture the more complex, tree-like homologies. Furthermore, we demonstrated the utility of our tool by analyzing a large number of metabolic pathways of *E. coli* and *S. cerevisiae*, thus revealing new biological insights into pathway evolution. These results in themselves are of interest and open the way to similar studies.

Future research may include extending the tool to more general network topologies, such as directed acyclic graphs, graphs with limited tree-width, and graphs that have simple cycle decompositions. Another open issue is to incorporate a variable scoring scheme, to *e.g.* represent affine gap penalties. We also propose to analyze hypergraphs: hyperedges can be used to represent reactions that involve several enzymes. One more interesting direction for future work may include incorporation of prior knowledge into the alignment of metabolic pathways and using the techniques presented in Chapter 3 in order to reduce the time complexity of the matching process.
Chapter 5

Functional Similarity Measure between Metabolic Genes

The MetaPathwayHunter tool presented in the previous chapter uses the EC classification to obtain an enzyme-to-enzyme similarity measure. However, EC — as well as other similarity measures that are based on manual biochemical annotations — suffers from a serious problem: missing, incomplete, or multiple annotations. Therefore, such similarity measures may cause inaccurate alignment results. For example, many enzymes in EC are annotated just by the main enzymatic activity class, such as [1. – . – .–], and thus two such nodes receive a perfect match score during the alignment. Another example is multiple annotations for a single gene (which is later translated to an enzyme) in Gene Ontology [14]. In this case, we need to somehow average the similarity measure over all annotations to get a final enzyme-to-enzyme similarity score which again causes loss in alignment precision.

This problem calls for a computational measure which can reflect the functional similarity between metabolic genes. First steps in this direction were made in [63] and [16] where the functional similarity was defined to be proportional to the distance between the corresponding enzymes in the metabolic network. In this chapter we propose a more comprehensive approach which is based on fluxes in the metabolic network and show that our measure outperforms the previously proposed ones.

The results reported in this chapter appear in [97].
5.1 Background

Mathematical modeling is used widely in systems biology to elucidate both cell activity as well as genes’ function and expression. Much of the work to date has attempted to establish measures for the similarity (or distance) between genes that are based on the topological properties of metabolic networks. Even though recent analyses have provided valuable insights regarding this issue [54, 91], topological characteristics alone (as devised by e.g. [63], [16]) offer only a static description of the properties of interest. On the other hand, accurate prediction of dynamic cell activity using kinetic models requires detailed information on the rates of enzyme activity which is rarely available; moreover, such analysis is usually limited to small-scale networks.

Fortunately, for metabolic networks, the use of stoichiometry and other sources of information can provide an added value over the topology of the underlying structure. Specifically, constraint-based models (CBMs) have emerged as a key method for studying such networks, permitting the large-scale analysis thereof. CBMs use genome-scale networks to predict steady-state metabolic activity, regardless of specific enzyme kinetics. In these models, stoichiometric, thermodynamic, flux capacity and possibly other constraints affect the space of attainable flux distributions.

In this chapter we employ constraint-based modeling to devise two effectively computable functional similarity measures between genes. The two measures employ large-scale in silico experiments, based on Flux Balance Analysis (FBA), that can be further validated in-vitro. Our first measure, the genetic response similarity (GRS) measure, is based on the similarity in metabolic network response to gene knockouts. The second measure, environmental response similarity (ERS), is based on similarity in the metabolic network activity across an array of various growth environments. These two measures reveal two complementary ways of defining the relation between gene $u$ with its surrounding: the GRS measure defines the effect of gene $u$ on its surroundings, whereas the ERS measure defines the effect of the surroundings on gene $u$.

To assess the veracity of the suggested measures, we validate them based on various biological data sources, including Gene Ontology (GO), phylogenetic profiling and gene expression measurements. The basic relation between metabolic fluxes and gene expression was previously studied and established both computationally (showing only a moderate correlation) as well as experimentally. Several studies [103, 104, 29, 93, 9] have shown that the expression patterns of enzyme coding genes are correlated with the flux patterns predicted by FBA. Here we extend these studies to look into ways of building upon the reported correlation between fluxes and expression, to construct efficient measures of functional similarity among metabolic genes. To this end, in contrast with the previous studies, we examine the relation between fluxes and expression while concomitantly controlling for correlations caused solely by the network’s topology.

Our comparison focuses on 750 metabolic genes of the yeast *Saccharomyces cerevisiae*. We
find that the ERS measure outperforms topological, conservation-based, and expression-based measures when testing for similarity with GO. Moreover, for many GO terms it is the only measure that succeeds to provide a significant result. On the other hand, the GRS measure shows only moderate results with only a few unique successes. We also find the correlation between model-based measures and co-expression to be statistically significant. However, we find GRS to be only moderately correlated with experimental data, whereas ERS exhibits a strong and significant correlation. Furthermore, this correlation remains so even after canceling the effect of the underlying (static) network topology. These results support the notion that a model-based environmental response similarity measure indeed captures the true functional similarity between metabolic genes.

Figure 5.1: A schematic illustration of two types of similarity measures between metabolic genes. (a) The Environmental response similarity (ERS) measure. Each element in vector $U(V)$ corresponds to the response of gene $U(V)$ to environment $i$. (b) The Genetic response similarity (GRS) measure. Each element in vector $U(V)$ corresponds to the response of flux $i$ to the knock-out of gene $U(V)$.

5.2 Computational Approach

5.2.1 Modeling Metabolism

Constraint based modeling provides a steady-state description of metabolic behavior. Flux Balance Analysis (FBA) [31, 58] is a particular constraint-based method which assumes that the network is regulated to maximize or minimize a certain cellular function, which is usually taken to be the organism’s growth rate. FBA has been demonstrated to be a very useful technique for the analysis of metabolic capabilities of cellular systems [90, 122]. It involves carrying out a steady
state analysis, using the stoichiometric matrix (as defined below) for the system in question. The system is assumed to be optimized with respect to functions such as maximization of biomass production or minimization of nutrient utilization; it is solved accordingly to obtain a steady state flux distribution, which is then used to interpret the metabolic capabilities of the system.

In FBA, the constraints imposed by stoichiometry in a chemical network at steady state are analogous to Kirchoff’s Second Law for the flow of currents in electric circuits [99], namely — for each of the \( M \) metabolites in a network the net sum of all production and consumption fluxes, weighted by their stoichiometric coefficients, is zero:

\[
\sum_{j=1}^{N} S_{ij} v_j = 0, \quad i = 1, \ldots, M
\]  

(5.1)

Here, \( S_{ij} \) is the element of the stoichiometric matrix \( S \) corresponding to the stoichiometric coefficient of metabolite \( i \) in reaction \( j \). The flux \( v_j \) is the rate of reaction \( j \) at steady state, and is the \( j \)-th component of an \( N \)-dimensional flux vector \( v \), where \( N \) is the total number of fluxes.

Additional constraints, including those pertaining to the availability of nutrients or to the maximal fluxes that can be supported by enzymatic pathways, can be introduced as the following inequalities:

\[
\alpha_j \leq v_j \leq \beta_j
\]  

(5.2)

For example, for a substrate uptake flux \( v_j \), one can set \( \alpha_j \) and \( \beta_j \) to be equal to the corresponding measured or imposed values. Eq. 5.2 can also be used to distinguish reversible and irreversible reactions, where \( \alpha_j = 0 \) for the latter.

A natural choice for an objective function in metabolic models of prokaryotes and simple eukaryotes is biomass production [90, 122], as it is reasonable to hypothesize that unicellular organisms have evolved towards maximal growth performance. This process is formalized by introducing a growth flux that transforms a linear combination of fundamental metabolic precursors into biomass.

The maximization of biomass production is implemented by defining an additional flux \( v_{gro} \) associated with cell growth. For this flux, the stoichiometric factors of the reactants are the experimentally known proportions \( c_i \) of metabolite precursors \( X_i \) contributing to biomass production [90]:

\[
c_1 X_1 + c_2 X_2 + \ldots + c_M X_M \xrightarrow{\text{gro}} \text{Biomass}
\]  

(5.3)

The search for the flux vector maximizing \( v_{gro} \) under the constraints of Eqs. 5.1 and 5.2 is
solved using the Simplex algorithm.

The theoretical basis of FBA is supported by several experiments. These include empirical validation of growth yield and flux predictions [90, 122], measurements of uptake rates around the optimum under various conditions [26], and results from large-scale gene deletion experiments [7].

For the stoichiometric analysis of the metabolic network of *S. cerevisiae*, we have used the reconstruction by Duarte, Herrgard, and Palsson [23]. The nodes of this network correspond to metabolic genes, and the edges correspond to the connections established by metabolic reactions. Two metabolic genes are connected if the corresponding enzymes share a common metabolite among their substrates or products. The list of metabolic reactions, and the 1060 (metabolites) by 1149 (fluxes) stoichiometric matrix (available at http://gcrg.ucsd.edu) were compiled using data from public databases and the literature. The 1149 reactions are associated with 750 genes. As in previous FBA formulations, we use inequalities (Eq. 5.2) to limit nutrient uptake and to implement reactions’ irreversibility. In addition to the 1149 internal reactions, we added to the model 116 uptake/excretion reactions, for each of the metabolites listed as “extracellular” in the basic model.

5.2.2 Model-based Similarity Measures for Metabolic Genes

In the context of the aforementioned motivation, we suggest two basic approaches for defining and measuring the similarity between metabolic genes: a genetic response similarity (GRS) measure and an environmental response similarity (ERS) measure. These are two complementary approaches, where the first reveals the effect of a genetic perturbation on the metabolic surrounding of a gene of interest, the other reveals the effect of the environmental perturbations on the gene of interest. A schematic illustration of both approaches can be seen in Fig. 5.1.

Genetic Response Similarity

Cellular response to a gene knockout involves rerouting of metabolic flux through alternative pathways and the utilization of isoenzymes [105, 111]. We hypothesize that similar metabolic responses to gene knockouts may provide evidence for similar metabolic functionality between genes. Based on this hypothesis, we define the GRS similarity measure between gene pairs as the similarity in the metabolic response following their knockout.

Predicting the metabolic response for gene knockouts is a more difficult task than predicting the metabolic state of wild-type strains. Gene deletion is commonly modeled by constraining the flux through the reactions associated with a given gene to zero, and applying FBA [90]. However, it turns out that the metabolic state of the knocked-out strain is not necessarily optimal in terms of growth rate, and thus in many cases FBA’s predictions are inaccurate. Instead, it was
hypothesized that the cell adapts to gene knockouts by minimizing the change in its metabolic state. Specifically, the Minimization of Metabolic Adjustment (MOMA) approach searches for a metabolic state for a knocked-out strain with minimal distance, under the L2 norm, from the flux distribution of the wild-type strain [105]. Recently, a new method called Regulatory On-Off Minimization (ROOM) was suggested to predict metabolic states following gene knockouts, and was shown to provide better predictions of knockout phenotypes [111]. ROOM aims to minimize the number of regulatory changes required for the adaptation by minimizing the number of significant flux changes between the metabolic states of the wild-type and knocked-out states (i.e. using the norm L0).

![Diagram of flux similarity model](image)

Figure 5.2: Schematic illustration of the proposed flux similarity model. $w$ stands for the optimal flux distributions on the wild-type metabolic network, $v_1$ stands for the optimal flux distribution on the metabolic networks with the first flux knocked-out, and $v_2$ stands for the optimal flux distribution on the metabolic networks with the second flux knocked-out.

A naive method for measuring the distance between the metabolic responses of two gene knockouts would be to simulate the knockout of each of them individually using ROOM, and then compute the distance between the obtained flux distributions. However, in many cases ROOM (like FBA and MOMA) provides multiple possible metabolic states for the knocked-out strain rather than a single solution. In these cases, it is not clear how to define the similarity measure between two genes.

To overcome this problem we define the GRS similarity measure as the minimal distance between the optimal ROOM solutions for the two genes. This is achieved by formulating a single optimization problem to find two ROOM solutions with minimal distance between them. The schematic illustration of our model is presented in Fig. 5.2.

Notably this formulation depends on the choice of a wild-type and thus we repeat our analysis for several different wild-types. Furthermore, since ROOM requires Mixed Integer Linear Programming (MILP) optimization which is NP-hard, we use a relaxed version of ROOM and — in addition — we use the L1 norm instead of L0. The distance between the two flux distributions of
the knocked-out strains is also minimized using the L1 norm.

The optimization problem is formulated as a LP problem as follows:

$$\min \|v_1 - v_2\|_{L1}$$

s.t.

$S \cdot v_1 = 0$;
$v_{\text{min}} \leq v_1 \leq v_{\text{max}}$;
$v_1[ko_1] = 0$, $ko_1 \in A_1$;
$v_2[ko_2] = 0$, $ko_2 \in A_2$;
$$\|w - v_1\|_{L1} = l_1;$$
$$\|w - v_2\|_{L1} = l_2;$$

(5.4)

where $w$ is the wild-type flux distribution, $A_1$ and $A_2$ are sets of reactions associated with the deleted genes, and $l_i$ ($i = 1, 2$) are the optimal solutions of a single optimization problem:

$$\min \|v - w\|_{L1}$$

s.t.

$S \cdot v = 0$;
$v_{\text{min}} \leq v \leq v_{\text{max}}$;
$v[ko_1] = 0$, $ko_1 \in A_1$;

(5.5)

Solving the above optimization problem we receive a measure of similarity between fluxes. The pseudocode of the procedure for computing GRS appears on page 70.

Throughout our study, we also examined the effect of excluding the isoenzymes from the analysis, as the model is uncapable to define which one of them is active. Notably, after exclusion of isoenzymes, the results obtained remain qualitatively similar across the entire analysis.

**Environment Response Similarity**

This measure aims to capture the similarity between the patterns of flux activity of two genes across a variety of growth media. To this end, we follow and extend the approach of [9], which studied the relation between the flux ranges of different reactions/genes and their regulation and conservation. Specifically, we compute genes’ activities across 100 randomly generated growth media, employing flux variability analysis: for each reaction we computed the maximal and minimal flux values attainable in the space of optimal flux distributions for growth conditions simulating 100 different growth media. The approach begins with determining the wild-type value of the
**Algorithm 7** ComputeGRS(): First, for each knocked-out gene the flux distribution is computed over the remaining fluxes. Then for each pair of genes the minimal distance (under $L_1$ norm) between the corresponding flux distributions is computed. When solving LP problems, $S$ is a stoichiometric matrix and $v_{min}, v_{max}$ limit nutrient uptake and define the reactions’ irreversibility.

**Output:** results - matrix `num_genes × num_genes` containing the distance between metabolic genes.

1: **procedure** ComputeGRS
2: Run FBA to maximize biomass (growth rate), obtain wild type flux distribution $w$
3: **for** each gene $g$ **do**
4: $A$ ← set of reactions associated with the deleted gene $g$
5: Compute knock-out flux distribution $v_g$ and a minimal distance from the wild-type distribution $l_g$ by solving the following LP problem:
6: min $\|v_g - w\|_{L_1}$
7: s.t. $S \cdot v_g = 0; \quad v_{min} \leq v_g \leq v_{max};$
8: $v[ko1] = 0, ko1 \in A;$
9: **end for**
10: **for** each gene $g_1 \neq g_1$ **do**
11: $A_1$ ← set of reactions associated with the gene $g_1$
12: $A_2$ ← set of reactions associated with the gene $g_2$
13: results$[g1][g2] = \text{dist}$ where \text{dist} is an objective function of the following LP problem:
14: min $\|v_{g1} - v_{g2}\|_{L_1}$
15: s.t. $S \cdot v_{g1} = 0; \quad v_{min} \leq v_{g1} \leq v_{max};$
16: $v_{g1}[ko1] = 0, ko1 \in A_1;$
17: $S \cdot v_{g2} = 0; \quad v_{min} \leq v_{g2} \leq v_{max};$
18: $v_{g2}[ko2] = 0, ko2 \in A_2;$
19: $\|w - v_{g1}\|_{L_1} = l_{g1}; \quad \|w - v_{g2}\|_{L_1} = l_{g2};$
20: **end for**
21: **end for**
22: **end procedure**
objective function by solving the base LP problem:

\[
\max c^T v \\
\text{s.t.} \\
S \cdot v = 0; \\
v_{\min} \leq v \leq v_{\max};
\]

From this solution the range of variability that can exist in each flux in the network due to alternate optimal solutions can be calculated through a series of LP problems wherein the value of the original objective is fixed and each reaction in the network is maximized and subsequently minimized to determine the feasible range of flux values for each reaction. Similar analysis has been used to identify bounds on the fluxes to further constrain the flux space for the identification of minimal reaction sets ([13]). The mathematical formulation of this approach is described below:

\[
\max v_i \\
\text{s.t.} \\
S \cdot v = 0; \\
v = z_{\text{obj}}; \\
v_{\min} \leq v \leq v_{\max};
\]

\[
\min v_i \\
\text{s.t.} \\
S \cdot v = 0; \\
v = z_{\text{obj}}; \\
v_{\min} \leq v \leq v_{\max};
\]

where \( z_{\text{obj}} \) is the value of the objective function calculated previously from 5.6, and \( n \) is the number of fluxes. The solution of the 2\( n \) LP problems outlined in 5.7 and 5.8 determines the upper and lower bounds of every reaction flux that will result in the same value for the original objective function.

Random growth media were generated by setting limiting values to the uptake reactions independently at random. With probability 0.5, the maximal uptake rate was set to 0, i.e. only excretion was allowed. Otherwise, uptake rate was limited to a value chosen uniformly at random in the range [0.01, 5], at a resolution of 0.01. A similar sampling method was used in [3]. In addition, to ensure sufficient variability between media, we switched between aerobic and anaerobic growth media with probability 0.5.

For each generated growth medium, we predicted which of the reactions are active, i.e., carry a non-zero metabolic flux (namely either its maximum or minimum flux values are different than
Active genes were denoted by '0' and nonactive ones by '1'. This way we created for each gene a binary vector of its activity across a series of generated media. We define the ERS measure as the normalized Hamming distance \[45\] between two binary vectors reflecting metabolic genes' activity. The normalized Hamming distance measures the degree of overlap between two sets of values, \(x\) and \(y\), and is computed as the fraction of unmatched nonzeros between \(x\) and \(y\) among all nonzeros of \(x\) and \(y\):

\[
\hat{h}(x, y) = \frac{x^T x + y^T y - 2x^T y}{n},
\]

where \(|x| = |x|^2 = |x||1\) is the number of nonzeros in an \(n\)-dimensional binary vector \(x\).

The pseudo-code of the entire procedure is presented on page 73.

5.2.3 Topology-based Measure for Metabolic Genes

As proposed in \[63\], the metabolic network structure can be used to calculate the network distance between genes. Let us define a pair of directly connected metabolic genes as separated by distance 1, and the network distance between genes \(X\) and \(Y\) to be the length of the shortest path from \(X\) to \(Y\) in the metabolic network. For sake of consistency, we call this measure a topology based similarity (TOBS) measure. While any metabolite can be used to establish connections between metabolic genes, the relationships established by the common metabolites and cofactors — such as ATP, water or hydrogen — are not likely to connect genes with similar metabolic functions. Hence, when compiling the metabolic network to this end, we consider a subset of metabolites which excludes the most highly connected metabolic species. An exclusion threshold was determined based on the connectivity of the resulting network. A total of the 10 most highly connected metabolites (ATP, ADP, AMP, CO\(_2\), H, H\(_2\)O, NADP, NADPH, phosphate and diphosphate), which compose 1% of all metabolites, and their mitochondrial and external analogs were excluded. Excluding up to the top 3% of all metabolites maintains the general trends described above.

5.2.4 Expression-based Measure for Metabolic Genes

We used Rosetta’s compendium dataset \[51\] which measures expression profiles of over 6200 \textit{S. cerevisiae} ORFs across 287 deletion strains and 13 chemical conditions. In addition, the dataset contains 63 negative control measurements comparing two independent cultures of the same strain. These were used to establish individual error models for each ORF, providing not only the raw intensity and the ratio measurement values for each experimental data point, but also a \(P\)-value evaluating the significance of change in expression level. The expression based similarity (EXBS)
Algorithm 8 ComputeERS(N): For each simulated medium flux variability analysis is applied in order to create an activity profile for each gene. Then the distance between the computed profiles is calculated.

**Input:** N - the number of required media.

**Output:** results - matrix num_genes \times num_genes containing the distance between metabolic genes.

```plaintext
1: procedure ComputeERS(N)
2:   for k=1..N do
3:     for each external flux f do
4:       with probability 0.5, set f = 0 otherwise f receives a random value chosen uniformly in the range [0.01, 5]
5:     end for
6:     Run FBA to maximize biomass (growth rate), obtain wild_growth_rate
7:     Add constraint: biomass \geq 0.9 * wild_growth_rate
8:     for i=1..num_fluxes do
9:       Run FBA to maximize flux i, obtain \( i_{\text{max}} \)
10:      Run FBA to minimize flux i, obtain \( i_{\text{min}} \)
11:     end for
12:     for each gene g do
13:       if for one of its related fluxes \( i_{\text{max}} = i_{\text{min}} = 0 \) then
14:         activity_vec[g][k] = 1
15:       else
16:         activity_vec[g][k] = 0
17:       end if
18:     end for
19:   end for
20:   for each gene g1 do
21:     for each gene g2 \neq g1 do
22:       results[g1][g2] = Hamming_distance(activity_vec[g1], activity_vec[g2])
23:     end for
24:   end for
25: end procedure
```
measure between ORFs \( X \) and \( Y \) was computed according to \( 1 - |\text{Spearman rank}(p_x, p_y)| \) where \( p_x \) and \( p_y \) are expression profile vectors of \( X \) and \( Y \), respectively, and the Spearman rank was calculated as in [89].

5.2.5 Phylogenetic Profiling Analysis

Ten sequenced fungal genomes (\( S. \) cerevisiae, \( C. \) albicans, \( C. \) glabrata, \( C. \) neoformans, \( D. \) hansenii, \( E. \) cuniculi, \( E. \) gossypii, \( K. \) lactis, \( S. \) pombe, \( Y. \) lipolytica) were used to construct phylogenetic profiles. The phylogenetic profile of a gene is a string of ones and zeros that encodes the presence or absence, respectively, of the gene in the corresponding genomes. We define a conservation based similarity (COBS) measure to be the similarity between phylogenetic profiles, computed using a normalized Hamming distance [45].

5.3 Results

We applied the proposed computational measures to the metabolic network model of \( S. \) cerevisiae by [23]. The model consists of 1060 metabolites and 1149 reactions (accounting for 750 genes). The obtained ERS and GRS measures were found to be significantly correlated (\( R^2 = 0.53 \), \( p \)-value \( = 3.2 \cdot 10^{-3} \)), testifying that indeed both measures capture the same overall signal. The remaining analysis provides evidence that these measures are indeed indicative of functional similarity and outperform the strictly topological measures.

5.3.1 Validating the Similarity Measures based on GO

To assess the accuracy of the ERS and GRS measures we compared them to the Gene Ontology (GO) functional annotations. Specifically, we expect an accurate similarity measure to have relatively high values for genes that are annotated with the same GO term, and low values for genes in different terms. In our analysis we used all non-redundant (i.e containing different genes) GO terms of sizes between 5 and 100. The overlap between these gene sets is quite low, as shown in Figure 5.3. For each such GO term, we computed the average distance between all genes annotated with this term. To assess the statistical significance of the average distance, we compared it to average distances obtained for 10000 random sets of genes, whose annotations were randomly shuffled while preserving the overall annotation distribution, obtaining an empirical \( p \)-value. The resulting \( p \)-values were further corrected for multiple testing of the many annotations via the false discovery rate (FDR) procedure [8].

We define a GO term to be consistent under some similarity measure if the resulting \( p \)-value (after FDR correction) for this term under this similarity measure is significant (\( \leq 0.05 \)).
Our results show that 86.5% and 18.7% of the GO terms are consistent under the ERS and GRS measures, respectively (Fig. 5.4). Interestingly, although the ERS provides better results overall, in some cases, only the GRS measure truly captures the functional similarity within some GO terms. For example, GRS finds the GO term 42724 corresponding to thiamin and derivative biosynthetic process to be consistent, while ERS does not. The results suggest that GRS outperforms ERS only for small GO terms (of size 5) where ERS does not receive a $p$-value significant enough to define a GO term as consistent. One putative reason for this may be the noisyness of the ERS measure, due to the large number of genes that tend to be active across many growth media. Comparing the ERS measure with other commonly used measures of functional similarity (Fig. 5.5 and Methods), we find that the ERS measure outperforms both: the EXBS and COBS measures, which obtain only 29.1% and 12.6% of consistent GO terms, respectively. Moreover, 82% and 96.5% of consistent terms found by EXBS and COBS measures were also found consistent by the ERS method. Using an alternative similarity measure (the Jaccard coefficient) between phylogenetic profiles provided similar results. Additionally, we tested the COBS measure with a different set of phylogenetic profiles, consisting of 17 higher eukaryotes (from NCBI’s HomoloGene’s database). Using this dataset, only 5.6% of the GO terms were found to be consistent under the COBS measure, testifying that conservation coherency is indeed much stronger among the closely related yeast genomes. As a next step, we compared the accuracy of the ERS and GRS measures to a measure obtained by considering only the topology of the network — TOBS (Fig. 5.5). We find that the ERS measure outperforms TOBS with 86.5% against only 63.9% rate of discovering consistent GO terms. ERS covers 82% and 96.5% while TOBS covers only 74% and 3.5% of consistent GO terms found by EXBS and COBS, respectively.

To gain further insights as to why ERS outperforms the simpler topological measure, it is
illustrative to examine the GO term 0006696 which corresponds to the process of ergosterol biosynthesis, for example. As shown in Fig. 5.6, ergosterol biosynthesis is carried out through a long, chain-like pathway, and hence the average distance between genes annotated with this term is significantly high (with a topological similarity p-value of 0.35). On the other hand, since these genes form an un-branched linear pathway, mass-balance constraint determines that all genes should either be coherently active or non-active. In this case, both the ERS and GRS measures show significant high similarity scores with p-values of $9.99 \cdot 10^{-5}$ and $1.7 \cdot 10^{-3}$, respectively. We note however that for this specific example, the expression similarity term is also relatively high, with a $p$-value of $1.8 \cdot 10^{-3}$.

Figure 5.4: A Venn diagram displaying the consistency of GO terms under the ERS and GRS measures.

Figure 5.5: A Venn diagram presenting the consistency of GO terms under ERS and topology-based similarity (TOBS) vs. the (a) expression-based similarity (EXBS) and (b) conservation-based similarity (COBS) measures.

Other cases where the topological similarity measure fails to identify true functional similarities relate to the identification and removal of currency metabolites. The removal of currency
metabolites (which are hubs in the network) is essential for the topological similarity measure to make any sense. Without the removal of these metabolites, the average distance between two genes is as low as 1.78 and only 1.3% of the GO terms are identified as consistent. However, the removal of currency metabolites may cause functionally related genes to be relatively far. For example, the genes annotated as involved in GO term 15698, corresponding to inorganic anion transport dissociate into four densely connected clusters in the network if the currency metabolite inorganic phosphate is removed.

Figure 5.6: The Ergosterol biosynthesis pathway. Each node (rectangle) represents an enzyme (except for the last one (ellipse) representing the final product — ergosterol). Each edge represents a metabolite which is produced by one enzyme and consumed by the following one in the pathway. Since ergosterol biosynthesis is carried out through a long, chain-like pathway, the average distance between genes annotated with this term is significantly high, while a mass-balance constraint determines that all genes should either be coherently active or non-active. Thus ERS outperforms the TOBS topological measure.

5.3.2 Validating the Similarity Measures based on Gene Expression Data

Similarity in gene expression patterns across multiple conditions is commonly used as indication of functional similarity [27, 116]. Specifically, this paradigm is further strengthened in the context of metabolic genes, whose expression is adjusted ‘just-in-time’ according to metabolic demands [127]. Notably, although similarity in expression is believed to be indicative of functional similarity, a comparison between the two only reveals a moderate correlation [106], with this claim further supported in the results shown in Fig. 5.5.

Measuring the correlation between the GRS and EXBS measures we observed (see Fig. 5.7 (a)) a moderate correlation ($R^2 = 0.38$ with a $p$-value of $2.1 \cdot 10^{-2}$). As for the ERS measure, we observe (see Fig. 5.7 (a)) that it exhibits a strong correlation with the expression similarity ($R^2 = 0.94$)
with a \( p \)-value of \( 1 \cdot 10^{-9} \). The correlations were obtained using a linear binning procedure [106] which averages one measure values over uniform intervals of the second measure. We note that our results regarding the correlation between ERS and expression similarity are in agreement with previous findings [103, 104, 29, 93, 9].

Measuring the topological similarity measure and expression similarity showed a weaker, but still strong correlation of \( R^2 = 0.78 \) (\( p \)-value = \( 6.6 \cdot 10^{-5} \)), demonstrating that genes closer to each other in the metabolic network tend to have, on average, higher level of co-expression (Fig. 5.7(b)), in agreement with the previous findings of [63].

Finally, we tested whether the ERS measure is advantageous over the TOBS measure, using a partial correlation test [61]. The partial correlation method quantifies the correlation between two variables whilst eliminating the effects of another variable on this relationship, namely network distance in our case. Our results show a significant partial correlation (\( R^2 = 0.65 \), with a \( p \)-value of \( 3.8 \cdot 10^{-6} \)) between ERS and similarity in expression levels. This result further supports the claim that the ERS similarity measure better captures the true functional similarity between genes compared to the TOBS topological measure. Furthermore, this result reaffirms FBA’s ability to accurately predict metabolic behavior across multiple conditions.

![Figure 5.7](image)

Figure 5.7: Correlation between co-expression levels (EXBS) and model-based (GRS and ERS) or topology based (TOBS) measures. The correlation is obtained by dividing the EXBS axis into uniform intervals and averaging the corresponding values of GRS, ERS and TOBS in each interval. (a) GRS/ERS measures. (b) TOBS measure.

5.4 Discussion

This study shows that metabolic network-based similarity measures between genes can go beyond previous measures that are based solely on network topology. We applied two schemes to compute this similarity: the genetic response similarity (GRS) scheme and the environmental response sim-
ilarity (ERS) scheme. While the former shows a fairly moderate correlation with the experimental results as well as a pretty modest ability for explicating GO terms, the latter provides a strong, statistically-significant measure. One possible explanation of this behavior may be that the ERS studies probe the natural wild type across a variety of media, whereas the GRS method does it in less natural strains and in a sole media. Another reason may be the more cumbersome computational method used in the GRS case, which is likely to add significant noise to the results obtained.

Furthermore, when examining the correlation with co-expression levels, one can observe that the GRS measure shows a certain decline as levels of EXBS approach 1. We believe that this phenomenon is driven by the nature of the GRS measure which is based on an underlying process of rerouting the metabolic fluxes through isoenzymes and alternative pathways. Recently, [55] have shown that in yeast most duplicate-associated backups involve genes that — on average — are not strongly co-expressed.

Notably, one cannot expect to find a 100% accuracy in finding consistent GO terms under the model-based measures as well as an absolute correlation between the model-based measures and gene co-expression. In essence, the fluxes predicted by the ERS measure across various growth media reflect a “wishful thinking” of an ideal system whose regulatory apparatus has developed with the sole optimization objective of maximizing growth. In this sense, the high levels of consistent terms (80-90%) and the high levels of correlations (0.8-0.9) found with the ERS measure in this study are truly striking.

Similarity in gene expression patterns across multiple conditions is commonly used as an indication of functional similarity. However, our results show that in many cases genes that are annotated with the same GO term are not expression coherent. Specifically, we find that only ~30% of the GO terms are composed of genes which are expression coherent. This lack of expression coherency may be the result of the complex interplay ongoing between metabolic and hierarchical regulation [117]. Remarkably, the ERS and GRS measures show significant high similarity values for 62.6% and 13% of the GO terms that are not expression coherent, showing their advantage over this traditional similarity measure.

One important problem that can be addressed in this context is that of functional prediction of gene annotation. It is well known that sequence similarity predicts rather well GO function annotations but fails to predict GO process annotation. In a similar vein, we computed the correlation between GO function and process annotations and sequence similarity of metabolic genes, using the measure of semantic similarity introduced by [94]. We observed a significant correlation between sequence similarity and GO functional annotations ($R^2 = 0.95$, $p$-value = $2.3 \cdot 10^{-5}$), while for process annotation the correlation was very low and insignificant ($R^2 = 0.4$, $p$-value = 0.2). Hence, quite obviously there is much room for new approaches for process annotation. Our study suggests that model-based, topology-based and expression measures can
contribute to the GO process annotation in a synergistic manner, with ERS having the largest potential contribution. Nevertheless, the goal of the method presented is not to provide functional annotation of new, unannotated genes, but rather to explore the functional relations between genes across the network, showing quite a few novel and interesting insights.

Finally, it is pertinent to consider the role of genomic and annotation information used in the reconstruction of the metabolic networks that are at the basis of our approach. We believe that one of the main ideas underlying the study of networks in systems biology is that one may find emergent network properties, i.e. new phenomena that were not explicit when constructing the network from its basic building blocks. The same idea is applied in this work: although genomic and annotation information have been used during the reconstruction of the metabolic network, our model is further based on considerable additional information, including the intrinsic network topology, the reactions stoichiometry, the growth media, and the mass balance and biomass maximization assumptions. All these transcribe together in a complex manner to reveal additional and different functional roles/annotations of the genes involved, as testified to by the results we report in this chapter. Specifically, one can note that our approach is essentially different than using similarity that is solely computed based on GO annotations (known also as semantic similarity). First, the latter is based on the partitioning of genes to groups/terms while this partition does not explicitly exist in the metabolic network. Furthermore, as we show by comparing to expression and conservation data, the functional similarity measures presented in this study outperform the metabolic network topology based similarity measure which is obviously closely related to the GO annotation.
Chapter 6

Conclusions

This thesis focuses on optimization problems derived from graph pattern matching which have applications to systems biology. We extended the exact subtree homeomorphism problem to an inexact one, developed new techniques to solve this problem for all possible kinds of trees so as to allow using this approach in biological applications such as alignment of metabolic pathways, and developed a new technique for devising a functional similarity measure between metabolic genes.

The optimization techniques described in this thesis can be viewed as complementary to traditional pattern matching, topological structure comparison, and similarity measuring methods. Our starting point was a classical subtree homeomorphism problem, which turned to be too strict when applying to real data. Thus, presenting a node-to-node similarity measure and turning subtree homeomorphism to an optimization problem was a promising direction. This new problem, termed Approximate Labeled Subtree Homeomorphism (ALSH), combines similarity measures with the topological distance between the matched trees to produce a single, comprehensive score expressing how close they are to each other. Besides the high modeling potential, this problem opened a wide range of computational challenges allowing to significantly improve the time complexity of the basic solution. These techniques are described in Chapter 2. Another related problem addressed in this thesis is the consistency of the resulting match, i.e. whether the found solution agrees with the expected one. We proposed to include prior knowledge about the desired result, termed as seeds, into the matching algorithm in order to ensure its consistency. The side-effect of using this knowledge was an additional complexity improvement of the entire family of matching algorithms, known as LCA-preserving matchings. These findings are presented in Chapter 3.

The developed methods can be used either in computational biology for comparing RNA secondary structures or matching phylogenetic trees, as well as in other areas such as semistructured databases and linguistics. However, the main application of interest was the analysis of metabolic
pathways, where the need arises in searching for homologues to a query pathway in a collection of known pathways, and of aligning two pathways to locate conserved pathway fragments. This application has proved to be very useful and capable of revealing new insights into pathway evolution, as can be seen in Chapter 4. The node-to-node similarity measure used here should reflect the functional similarity between the enzymes combining the aligned pathways. Thus, the functional similarity of metabolic genes naturally drew our attention. Chapter 5 describes a new computational approach for revealing such a similarity using metabolic fluxes in reconstructed metabolic networks. This approach is shown to be the best to date compared to other computational approaches and it is consistent with known biologically-based measures.

One conclusion of this thesis is that many practical problems in systems biology may be seen as constrained combinatorial optimization problems. These problems can be solved quite efficiently using the techniques presented herein. In addition, this thesis shows that a node-to-node similarity measure is the core of any matching approach and should be picked carefully. Such a measure should be both informative and at a high resolution. An example is the EC classification which suffers from missing or incomplete annotations. Thus, the computational measure presented in Chapter 5 can provide a better result when aligning metabolic pathways.

This thesis covers some of the major problems which may arise in matching objects modeled by labeled graphs. But one may still ask whether the approach described herein is more useful than other pattern matching techniques? From the point of view of a user, there is one major advantage in using the proposed methods — they provide an optimal solution within a low time complexity. Other known techniques — such as heuristics, approximations, or branch-and-bound exhaustive search — are either not optimal or applicable only to small problems. Therefore, our approach is uniquely useful for an interesting subset of practical applications.

By studying constrained combinatorial optimization techniques, this thesis has extended the scope of classical pattern matching in systems biology. At the same time, it demonstrated that the potential of this field of optimization is far from being exhausted. One open problem that remains is whether there exists a polynomial time algorithm for finding subgraph isomorphism/homeomorphism in directed acyclic graphs. Future work may include extensions of ALSH to more complex topologies, such as bounded tree-width graphs, planar graphs, and finally directed acyclic graphs. Another interesting extension of the current study may be the development of seeded tree alignment for other families of tree matching algorithms, such as edit distance, and the incorporation of seeded alignment into the MPH tool. In addition, the study on how different similarity measures affect the resulting match when aligning metabolic pathways seems to be a promising direction.

New advances in biochemical technology makes it possible to reconstruct new metabolic networks at a relatively low cost. These new models will not only contain more reactions, but will
also have to take into account dependencies with other systems such as signaling and regulatory networks. New inter-system pathways will become available for analysis as well as new metabolic data such as enzyme rates, enzymes and metabolites concentration and more. Therefore, developing improved methods that can cope with the technological challenges are needed. These methods should include more complex topological structures as well as more precise similarity measures, which are based on enhanced complex biochemical systems.
Bibliography


APPENDIX A:
The MetaPathwayHunter Manual

User Guide

Input Form

The first screen, the user receives when running the MPH tool is the Input Form:

![Figure A.1: The input form.](image)

The Input Form consists of three main sections — Pattern, Text, Options — and the “Run ALSH” button which runs the tools with the provided inputs. The pattern section is used as a graph editor for drawing or loading the pattern graph to the MPH. On the left of the editing canvas there is a toolbar with the following buttons:

1. New Graph - Used to clean the editing canvas.
II. Load Graph - Used to load an existing graph file (*.grp).

III. Save Graph - Used for saving the graph drawn on the canvas.

IV. Add New Node - Used for adding a new node to the canvas. After clicking this button the mouse cursor will be changed to an oval which can be placed anywhere on the canvas. When the new node is placed, an input dialog will appear on the screen to allow the user to enter an enzyme EC number (in the form of a.b.c.d). If the number the user have entered is valid, a new node labeled with this number will appear on the canvas, and the mouse cursor will change back to normal. Otherwise, an error message will appear, notifying the user that the number they have entered is invalid. The user also has an option to change his mind and not to add a new node by clicking the 'Add New Node' button again — in this case the cursor will change back to normal immediately.

V. Add New Edge - Used for adding a new edge to the board. After clicking this button the mouse cursor will be changed to a green arrow with a dot on its tail. The user has to choose a “from” node for the new edge by clicking on it. The mouse cursor image will change again — this time to a green arrow with a dot on its head. The user has to choose a “to” node for the new edge by clicking on it. When the edge is placed the mouse cursor image will change back to normal. The user also has an option to change his mind and not to add a new edge by clicking the 'Add New Edge' button again.

VI. Remove Node - Used for removing a node from the board. After clicking this button the mouse cursor will be changed to an crossed oval. The user has to choose a node on the canvas to be removed and click on it. The node will be deleted from the canvas, and the mouse cursor image will change back to normal. The user also has an option to change his mind and not to remove the node, by clicking the 'Remove Node' button again.

VII. Remove Edge - Used for removing an edge from the board. After clicking this button the mouse cursor will be changed to a red arrow with a dot on its tail. The user has to choose a “from” node of the edge to be removed and click on it, the mouse cursor image will change again, this time to a red arrow with a dot on its head. The user has to choose a “to” node of the edge to be removed and click on it. The selected edge will be deleted from the canvas, and the mouse cursor image will change back to normal. The user also has an option to change his mind and not to remove the node, by clicking the 'Remove Edge' button again.

The Text section of the Input Form consists of two buttons and a text-field. The MPH tool can match the pattern graph drawn on the editor canvas to a single text graph file or to a directory of text graph files. The user has an option of selecting a single text graph file by clicking on the
left button. A browsing dialog will appear, allowing to choose a single graph file (*.grp). If the user prefers to select a directory of text graph files, he may click on the right button. A browsing dialog will appear, allowing to choose a directory. The MPH tool will load only the graph files (*.grp) in this directory. Once the graph files/s are selected, the user’s choice will appear on the text-field.

![Figure A.2: The file browser for the Text section.](image)

The Options section consists of a combo-box for choosing the graph type and a button for selecting the deletion score. The default graph type selected is “undirected graphs”. In order to change the graph type, the user should click on the combo-box and choose another graph type. If the “undirected graphs” type is selected, the tool ignores the direction of the edges in the text and pattern graphs although they appear both on the canvas and in the text files.

The default deletion score is $-2.0$. This score can be changed by clicking on the “deletion score” button. The input dialog will appear allowing to enter a new deletion score — the deletion score must be negative. If the entered score is valid, it will appear in the corresponding text area, otherwise, an error dialog will appear.

Once the user have finished filling the input form, he/she may click the ‘Run ALSH’ button. If the form is not filled properly, an error dialog will appear describing the missing details. If all the details are complete, a progress-bar will appear, reflecting the progress of calculating the ALSH on the chosen graphs.
Results Form

When the computation is done (and the progress bar reaches its end), the results table appears on the screen.

The results table consists of three columns: (1.) Text File Name - The name of the text graph file used in the match; (2.) Match Score - The score of the obtained alignment; (3.) P-value - The significance for the match.

The P-values are empirical and are calculated using a set of random text graph files. The MPH tool searches for a sub-directory named “random” under the directory of the chosen text files. If such a directory does not exist, the default random directory will be used. For example, if the user choose to run a pattern against the E.coli database located in ./MPH/ecoli/, the sub directory ./MPH/ecoli/random/ containing random text files designed for E.coli will be used to
calculate the P-values.

Clicking on one of the columns header will sort the results in the table according to the selected column. For example, clicking on the ‘score’ header will sort the results by match score, descending. Clicking on one of the table rows (representing a single match) will launch a ‘Match View’ window for the chosen match.

Alignment Form

The Alignment Form window is used for presenting the aligned graphs for a chosen match.

![Alignment Form](image)

Figure A.5: The alignment form showing both the pattern and the text.

The white nodes represent the text proteins and the green nodes represent the pattern proteins. Black labels placed on top of a text node represent the text enzyme EC number and green labels placed under a pattern node represent the pattern enzyme EC number. The graphs are drawn according to the coordinates of the text graph file, if such provided. Otherwise, the nodes are placed in a random manner on the canvas. Nodes pinned together with a thumbtack represent a node-to-node match between the text and the pattern’s enzymes. The size of the pattern’s nodes represents the node-to-node match score — the bigger the pattern node, the higher is the match score.
score. Pattern graph nodes with the same size as the text nodes represent a perfect match.

Holding the mouse cursor over a node will result in a tool-tip box popping out of the node, describing the relevant details such as the enzymes names and the match score.

Figure A.6: The details on node-to-node match.

At the bottom of the window there are two check-boxes. These check-boxes let the user choose which of the layers — pattern, text, or both — are drawn on the output canvas. The user may choose to see only the text graph layer, only the pattern graph layer, or both layers together (the default mode).

Figure A.7: The alignment form showing only the pattern.
Implementation Details

The high level design of the prototype tool is presented in Figure A.8. Each object is represented as a square and arrows represent the data flow between objects. The input consists of the text file, pattern file and score file and the output contains match files, text file and pattern file in html format ready for visualization.

Figure A.8: The high level design of the prototype tool.
APPENDIX B:
Studying Regulatory Mechanisms in Yeast using Environmental Conditions
Similarities and differences of gene expression in yeast stress conditions

Oleg Rokhlenko¹,*, Ydo Wexler¹ and Zohar Yakhini¹,²
¹Technion-Israel Institute of Technology, Department of Computer Science, Haifa 32000, Israel and
²Agilent Laboratories, Palo Alto, CA, USA

ABSTRACT

Motivation and Methods: All living organisms and the survival of all cells critically depend on their ability to sense and quickly adapt to changes in the environment and to other stress conditions. We study stress response mechanisms in Saccharomyces cerevisiae by identifying genes that, according to very stringent criteria, have persistent co-expression under a variety of stress conditions. This is enabled through a fast clique search method applied to the intersection of several co-expression graphs calculated over the data of Gasch et al. This method exploits the topological characteristics of these graphs.

Results: We observe cliques in the intersection graphs that are much larger than expected under a null model of changing gene identities for different stress conditions but maintaining the co-expression topology within each one. Persistent cliques are analyzed to identify enriched function as well as enriched regulation by a small number of TFs. These TFs, therefore, characterize a universal and persistent reaction to stress response. We further demonstrate that the vertices (genes) of many cliques in the intersection graphs are co-localized in the yeast genome, to a degree far beyond the random expectation. Co-localization can hypothetically contribute to a quick co-ordinated response. We propose the use of persistent cliques in further study of properties of co-regulation.

Supplementary information: http://www.cs.technion.ac.il/~olegro/stress.html
Contact: olegro@cs.technion.ac.il

1 INTRODUCTION

The mechanisms that control and regulate cellular processes in living organisms are complex and involve several types of control, monitoring and activation/de-activation modules. These are only partially understood and the subject of continuous efforts to better elucidate. Model systems, such as Saccharomyces cerevisiae, play an important role in this study. Some of the components of the mechanisms that control gene expression in yeast are, in fact, known and can even be reproduced or manipulated in the laboratory.

All living organisms and the survival of all cells critically depend on their ability to sense alterations in the environment and then respond promptly and adequately to new situations through the induction of protective stress responses. Yeast, as well as other organisms, employ a concerted response to external stress conditions. The genomics of stress response in S. cerevisiae has been extensively studied using a variety of experimental and computational techniques. In Ruis and Schuller (1995) the authors review three mechanisms of stress response in Saccharomyces cerevisiae—the positive transcriptional control activated by heat shock elements, stress response elements and AP-1 responsive elements. They identify yeast genes with a universal stress response as well as genes with a more specific reaction profile. In a break-through application of a high-throughput approach, Gasch et al. (2000) use expression profiling with microarrays to measure the changes, as a function of time, of almost all yeast genes, as a result of the exposure to a variety of stress conditions. They observe that a large set of genes (~900) show drastic response to most of the studied conditions. They also study the correlation between the response patterns of genes in single stress conditions by using clustering techniques. In this article we study the sets of genes that seem to be persistently and strongly co-ordinated as part of the stress response mechanism, not restricted to a single specific condition.

For every stress condition we define the co-expression graph to be an undirected graph whose vertices correspond to genes, and the vertices of two genes are connected by an edge if their expression profiles are sufficiently correlated. Namely, the p-value of the Pearson correlation between the expression patterns of the two genes is statistically significant (p-value < 0.01). Two genes are said to be co-co-expressed in stress conditions A and B if their expression patterns in both time-courses correlate; alternatively—if they have an edge connecting them in both co-expression graphs. The k-stress persistence graphs (k-pers) are the intersection graphs of sets of k co-expression graphs. By studying cliques in k-pers graphs we impose very stringent conditions of co-ordination on sets of genes. They must all be highly correlated with each other in all conditions under consideration.

The persistence of co-expression in different organisms was studied by Bergmann et al. (2004) who compared the expression profiles of six organisms. They found that co-expression is often conserved among organisms although the contribution of sets of genes to the overall expression varies.

In yeast the activation of stress response has been associated with the activity of a small number of transcription factors (TFs) that regulate complex expression patterns of a large set of genes. In Pilpel et al. (2001) the authors study the joint effect of TFs on gene expression in yeast, developing a framework for understanding the combinatorics of transcription regulation and observe that a small number of TFs regulate response to stress. Hvidsten et al. (2005) develop a rule-base mechanism for predicting stress related co-expression. In this study we used TF to mRNA association data (Harbison et al., 2004) to identify persistent cliques with enriched association to very few TFs. We also find persistent cliques to be functionally enriched using GO-term analysis.

*To whom correspondence should be addressed.
Finally, we observe an interesting relationship between persistent cliques and the genomic location of member genes. This significant proximity between genes that are persistently co-expressed under several stress conditions might indicate that the cell is using genomic proximity to facilitate prompt co-activation as response to stress.

The structure of this article is as follows: We start in Section 2 with some details regarding the expression data used, properties of the co-expression graphs, and the relationship between the different stress conditions. Then, Section 3 presents our approach and the main results. Finally, Section 4 gives a comprehensive description of the computational and statistical methods used.

2 DATA AND CO-CO-EXPRESSIOIN GRAPHS

In our analysis we used the expression dataset from Brown’s group (Gasch et al., 2000) containing 173 environmental perturbations (stresses) for *S. cerevisiae*. We considered only conditions with time series of length 5 or more resulting in 19 sets (CAR1, CAR2, CT1, CT2, DIAMIDE, DIAUXIC SHIFT, DTI, DT2, H2O2, 20MIN HEAT, HEAT 37-25, HEAT SHOCK1, HEAT SHOCK2, MENADIONE, NITROGEN DEPLETION, SORBITOL, SORBITOL 29-33, YPD1, YPD2). We removed genes with >25% missing expression values resulting in 6151 genes, and completed the remaining absent values in the data (less than 2%) with average expression levels of the relevant time series.

To investigate the network structure revealed for the stress conditions, we used a well-established topology property of connectivity distribution of a co-expression graph. We found that under all stress condition the obtained co-expression graph is power-law distributed, namely for \( k \) being the number of edges of a particular gene, the distribution is \( n(k) \propto k^{-\gamma} \) with exponent \( \gamma = 1.2 - 1.8 \). We note that in order to obtain the connectivity distribution we used a standard logarithmic binning. The boundaries of the bins were powers of two, and we counted the number of genes between two boundaries and normalized by the bin width. We applied the linear fit to the log values of the bin centers against the normalized counts. These findings are consistent with the ones of Bergmann et al. (2004) and extend them to hold under different stress conditions.

To study the relationship of the different stress conditions we constructed co-co-expression graphs for each pair of conditions. We then performed hierarchical clustering using average-linkage neighbor joining (NJ) (Saitou and Nei, 1987), considering two similarity measures: number of edges in each graph and a correlation computed for all edges.

The tree constructed using the measure of number of edges in the co-co-expression graphs is illustrated in Figure 1. The other measure of correlation for all edges in the graphs was computed as follows. For each stress condition, we computed a co-expression vector where each entry corresponds to the co-expression of a pair of genes under this stress condition. Perfect identity between two stress conditions is achieved when each pair of genes is co-expressed identically under both conditions. The correlation was computed via standard linear-regression method where the data was divided into 10 bins on one of the stress conditions, while averaged on the other condition. For symmetry, we also computed the correlation coefficients when binning on the other stress condition, and used the average of the two coefficients as the similarity measure. More details regarding these computations appear in Section 4.4. The results are shown in Figure 2. To verify the significance of the correlations, we ran 100 random tests for each pair of stress conditions. In each test, the topology of the co-expression graphs was maintained by shuffling genes in one of the conditions. The average correlation in these random experiments for all pairs of conditions was 0.029 with standard deviation of 0.011. Figure 3 shows the correlation between the two stress conditions HEAT SHOCK2 and DIAMIDE after binning on each of the conditions, as well as the best correlation from 100 experiments.
when shuffling the genes in the vector that corresponds to the expression levels under DIAMIDE stress condition.

The two trees constructed by hierarchical clustering agree on most relationships between the stress conditions. However, some of the results are not conclusive and two conditions that appear to behave similarly with one measure may appear distant when using the other (e.g. DDT1 and DTT2). Based on these results, we reduced the number of stress conditions to 12, by eliminating 7 of the original 19 conditions. First, one experiment of every condition under which two experiments were conducted was removed (HEAT SHOCK2, CAR2 and CT2). This reduction was motivated by the relative proximity of each such pair of experiments in the NJ tree. The only exception is the pair of DTT experiments which did not exhibit similar expression profile, and thus were both removed.

In addition, the two non-stress experiments, YPD1 and YPD2 were excluded and used only as reference profiles (Section 3).

3 APPROACH AND RESULTS

Here we present our approach and describe the obtained results.

3.1 Cliques

Genes with persistent co-expression across several stress-conditions are of special interest as they are likely to be involved in a universal stress response mechanism. Studying graphs of genes with persistent behavior is enabled through analyzing cliques in the k-persistence graphs.

A clique of a graph $G$ is defined as a complete subgraph of $G$. In the co-co-expression graph cliques correspond to groups of genes with highly similar expression under several stress conditions. Naturally, we are interested in large cliques and in this paper confine our study to maximal cliques (e.g. cliques of maximum size) and cliques of size close to maximum.

We extracted maximal cliques from all k-stress persistence graphs for $2 \leq k \leq 12$, and examined their properties. First, we analyzed the typical clique size for every number of stresses, as shown in Figure 4a. Surprisingly, we found extremely large cliques even when the number of stresses considered is large. For example, the maximal clique size in 5-stress persistence graphs was 27 for
the set of 5 stresses—NITROGEN DEPLETION, HEAT SHOCK1, 20MIN HEAT, DIAUXIC SHIFT and DIAMIDE—and for 3-stress persistence graphs was 53 for the set SORBITOL 29-33, HEAT SHOCK1 and DIAMIDE. Such large cliques indicate that genes tend to behave similarly under several stress conditions much more than expected by random. To show this, we examine the distribution of the size of maximal cliques in k-stress persistence graphs for $k = 2, 3$, where the genes were shuffled before intersecting the co-expression graphs. This distribution, which is plotted in Figure 4b, was taken from 100 random graphs for each set of stress-conditions, which amounts to 6600 graphs for $k = 2$ and 22000 for $k = 3$.

Some genes that are involved in fundamental processes are co-expressed even when not under stress. This co-expression may be important for the basic functions regardless of the existence of stress. Thus, to separate cliques of genes that are co-expressed everywhere from these introduced only under certain stress conditions, we calculated the percentage of pairs of genes in each clique that are connected by an edge in expression profile under regular conditions taken from Spellman et al. (1998). Figure 5 plots the distribution of percentage of conserved edges when there is no stress. Note that most cliques are highly conserved and to study regulation under stress we therefore want to focus on the cliques that are not conserved as explained in Section 3.2.3. The full table with the cliques and the percentage of edges that appear when not under stress is available in the Supplementary Data. We ran an identical analysis using expression levels reported in Gasch et al. (2000) for non-stress experiments YPD1 and YPD2 and received similar results.

### 3.2 Enrichment

Cliques in k-persistence graphs that exhibit significant enrichment levels to some feature can indicate a connection between this feature and a common process in which genes of the enriched clique participate. In addition, enriched cliques can be used as a tool to associate or link genes in the clique, for which no information is known.

We study the enrichment of two important types of features—transcription regulation factors, which are known to regulate certain genes, and Gene-Ontology (GO) annotations, which, associate genes to functions or processes, and provide a biologically interesting example, that incorporates enrichment for these two types of features.

#### 3.2.1 Transcription factors

The response of *S. cerevisiae* to diverse stress conditions is hypothesized to be regulated by a small number of transcription regulation factors (TFs) (Hvidsten et al., 2005; Pilpel et al., 2001). Therefore, these transcription factors are expected to regulate a relatively large number of genes under several stress conditions. These genes should furthermore be persistently co-expressed.

To examine the hypothesis we analyzed the TF enrichment in cliques of k-stress persistence graphs. TF with targets enriched in a certain clique is assumed to be driving the co-expression. Therefore, enriched TFs are driving universal stress response. This approach is complementary to that of Hvidsten et al. (2005) who developed a mechanism for predicting an expression pattern for given transcription factor binding sites present in the promoter region of a gene. Figure 6 shows the number of cliques enriched by different transcription factors as a function of the number of stress conditions in the k-persistence graphs. Our findings show that cliques are enriched only for six TFs (FHL1, RAP1, ABF1, SFP1, PAC and mRRPE) out of 113 taken from the data of Hvidsten et al. (2005), Pilpel et al. (2001) when setting a statistical threshold of $10^{-6}$, in agreement with previous results. When raising the threshold to $10^{-8}$, only four more TFs (YAP5, SNT2, PDR1 and YDR026c) are enriched in the cliques. The full table with all cliques enriched with TFs is available in the Supplementary Data.

We examined also the enrichment of cliques to pairs of TFs. As expected, significant enrichment ($p$-value $< 10^{-5}$) was observed only for a few pairs: (RAP1,FHL1), (RAP1,SFP1), (FHL1,SFP1), (PAC,ABF1), (PAC,mRRPE) and (ABF1,mRRPE). Note that all pairs consist only of the six TFs reported to be enriched by themselves. Moreover, this set divides into two triplets, which appear to be synergistic. These results partially support the results of Pilpel et al. (2001), where PAC and mRRPE are reported to be synergistic. We suggest that similar relations exist for the rest of the pairs as well.

#### 3.2.2 GO annotation

Gene-Ontology annotations help in classifying genes according to their function and the processes they participate in, and in interpreting experimental results. Not all genes have been associated with a GO-term. Actually, in *S. cerevisiae* there are 599 unannotated genes. We suggest the use of cliques that are significantly enriched with a specific GO-term but also contain unannotated genes as a tool to help annotate these genes.
We computed the enrichment of cliques with the different GO-terms, and like in TF enrichment, found that the cliques are enriched only with 19 GO-terms (see details in Supplementary Data).

### 4.1 Cliques

A clique in a graph \( G = (V, E) \) is defined as a complete subgraph of \( G \), namely, a subset \( V' \subseteq V \) such that for each pair of vertices \( v_1, v_2 \in V' \) the edge \( e = (v_1, v_2) \) is in \( E \). In general, the problem of finding the maximum clique for a given graph is NP-hard (Karp, 1972), and usually heuristics are used to solve this problem.

To verify that this co-expression clique is indeed a result of stress and such co-expression does not occur without stress, we counted the number of edges in the clique which also appear in a cell-cycle expression experiment (Spellman et al., 1998) and found that \( \leq 55\% \) of the edges are conserved in this experiment.

Although this large clique is persistent in 5-stress conditions, the co-expressions of the genes in the clique are a response to these specific five conditions: when adding, for example, the stress condition CT1 the maximal clique in the 6-stress persistence graph is only of size 6.

### 3.3 Genomic distance and co-expression

Previous works indicated the correlation between genomic proximity and genes that participate in the same metabolic pathways (Hurst et al., 2004; Overbeek et al., 1999). We analyze the \( k \)-stress persistence graphs to study the relationship between co-expression and genomic proximity and show that not only that genes which are co-expressed tend to be closer on the genome, but that their proximity is higher in graphs that represent greater persistence (i.e. large \( k \)-values). The distribution of genomic distance of genes connected by an edge as a function of the number of stress conditions is shown in Figure 8a, and the evidently decreasing average distance in Figure 8b. This average is low even for 2-stress persistence when compared with the expected random distance between genes on the same chromosome which is 378,350 bp. In addition to the distance of genes, measured only for pairs on the same chromosome, we also computed the fraction of genes connected by an edge that are in fact on the same chromosome. As shown in Figure 8c this value increases as the number of stress conditions considered is larger. Furthermore, the expected random fraction of 7.54%, which is depicted for reference, is lower than the fractions observed for the co-co-expression graphs.

### 4. Computational and Statistical Methods

#### 4.1 Cliques

A clique in a graph \( G = (V, E) \) is defined as a complete subgraph of \( G \), namely, a subset \( V' \subseteq V \) such that for each pair of vertices \( v_1, v_2 \in V' \) the edge \( e = (v_1, v_2) \) is in \( E \). In general, the problem of finding the maximum clique for a given graph is NP-hard (Karp, 1972), and usually heuristics are used to solve this problem. Unfortunately, all known methods work quite poorly on large graphs. We propose to use the special connectivity distribution of the co-expression graphs to efficiently find maximal cliques. The power-law distribution property helps to prune the search space drastically by first finding a trivial lower bound on the size of a clique and then iteratively discarding nodes with degrees less than this bound.

To efficiently solve the max-clique problem we employ the search algorithm Depth-First Branch and Bound (DFBnB). Given a group of vertices \( V \), we decide on some order \( v_1, \ldots, v_n \). At each depth \( i \) in the search tree we decide whether to include or exclude the vertex \( v_i \) from the chosen subset of vertices. We refer to the vertices that are included in the chosen subset as the included set, \( I \), and to the excluded vertices as the excluded set, \( X \). The rest of the \( n - m \) (undecided) vertices at that point are called the free set, \( F \).

Each node at the lowest level \( n \) defines a unique candidate subset \( C \subseteq V \), according to the vertices that are in the included set (\( C = I \).
Target nodes are those for which $C$ is a clique and the score of each target node is defined as the size of the clique corresponding to that node ($|C|$). For each node $N$ we define $g(N)$—the size of the included set $I$, $h(N)$—a heuristic estimation (upper bound) on the number of vertices that will be added from the free set in order for the subset to become a clique; $f(N) = g(N) + h(N)$—an upper bound on the clique size which can be obtained by adding vertices from the free set to the current included set. The DFBnB algorithm will search for a target node (clique) that maximizes $f(N)$, thereby designating a maximal clique. Owing to the power-law nature of the expression networks which are the input graphs, strong pruning is conducted via a simple lower bound on the maximal clique’s size and recursively removing all nodes which do not have a corresponding degree in the graph. This pruning allows us to handle and quickly find maximal cliques in large graphs containing millions of nodes as the common $k$-stress persistence graphs.

The algorithm is described in details in Figure 9.

### 4.2 Enrichment

In order to evaluate the enrichment level of cliques to a specific feature (e.g., certain transcription regulation factor or GO term), we use the Hyper-Geometric (HG) distribution. This measures the probability that a subset $S$ of $n$ independent objects from a larger set of $N$ objects is enriched with a given feature $f$. Namely, that at least $k$ objects in $S$ have feature $f$, while this feature is associated with $K$ out of $N$ objects in the entire set. Formally

$$P_{enrich} = \sum_{i=k}^{n} \binom{n}{i} \frac{\binom{N-n}{k-i}}{\binom{N}{k}}$$

Since we examine the enrichment levels of multiple cliques for each feature, we correct the $p$-values of the HG test by multiplying it with the number of cliques tested.

### 4.3 Co-location

The genomic distance of $k$-stress persistence is defined to be the average genomic distance over all edges in all $k$-stress persistence graphs. This distance is measured only on edges which connect pairs of genes on the same chromosome.

**Algorithm 1:** \texttt{FindMaxClique}(G)

**Input:** Graph $G$, lower bound $l$ on clique size.

**Output:** The maximal clique $C$ in $G$.

Order the vertices $v_1, \ldots, v_n$;

$max = l$; $i = 1$; $C \leftarrow \emptyset$; $I \leftarrow \emptyset$; $X \leftarrow \emptyset$; $F \leftarrow V$;

while $i > 0$ do

if $i = n$ then

If $max < |I|$ then $max = |I|$; $C \leftarrow I$;

else

if $f(I \cup v_i, F \setminus v_i) > max$ then

$I \leftarrow I \cup v_i$; $F \leftarrow F \setminus v_i$; $i \leftarrow i + 1$;

else

if $f(I, F \setminus v_i) > max$ then

$X \leftarrow X \cup v_i$; $F \leftarrow F \setminus v_i$; $i \leftarrow i + 1$;

else

$F \leftarrow F \cup v_{i-1}$; $I \leftarrow I \setminus v_{i-1}$;

$X \leftarrow X \setminus v_{i-1}$; $i \leftarrow i - 1$;

end

end

end

return $C$;

The expected random distance between genes on the same chromosome is 378 350 b, and the overall fraction of genes on the same chromosome in \textit{S.cerevisiae} is 7.54%. To verify that the expected distance is maintained for graphs with topology such as the $k$-stress persistence graphs, we shuffled the genes names 100 times for each graph, and measured the average distance. In the same manner we also measured the random fraction of genes connected by an edge which are on the same chromosome.

### 4.4 Correlation by binning

The dispersion of the data is an intrinsic problem in biology. The noise in the measurements and poor annotation increase the inherent variability. To ascertain underlying data trends, we use logarithmic
and linear binning procedures. Given a set of points \((x, y)\) and the axis on which we perform the binning (without loss of generality we explain here for \(Y\) axis), we divide axis \(Y\) into \(M\) intervals (bins). Linear binning divides axis \(Y\) uniformly, namely, each bin size \(D = (b - a)/M\) where \(a\) and \(b\) are the minimal and maximal values on the binned axis, respectively, and \(\phi = [(y_i - a)/\Delta]\) is the bin index we assign to a data point \((x_i, y_i)\). For each bin \(\phi\) we create a point \((x_{\phi}, y_{\phi})\) where \(y_{\phi} = a + D(\phi + 1)\) and \(x_{\phi}\) is the average of all \(x_i\) in this bin. Logarithmic binning is similar to linear binning in all but the bin size \(D = (1/M) \log (ab)\) and \(\phi = \log\left(\frac{y}{a}\right)/D\). As before, for each bin \(\phi\) we create a point \((x_{\phi}, y_{\phi})\) where \(y_{\phi} = ae^{D(\phi+1)}\) and \(x_{\phi}\) is the average of all \(x_i\) in this bin. The advantage of both procedures is in elevated degree of noise reduction but the logarithmic binning is usually used to correct the skewed nature of the scale-free distribution (Albert et al., 2000).

ACKNOWLEDGEMENTS

The authors thank Doron Lipson for helping with the clique finding program design and implementation, Eran Eden for supplying references on transcription factors data and Yonatan Aumann for discussions and references on genomic distance and co-expression.

REFERENCES


הпуска את תופעת המיתוג של המנהלים והאנזימים המבוססים על אנג'ה. המבוססים על אנג'ה, הפועלים עם תיאוריה ומענה licitation. באחת על אנזימים באמצעות חומצות DNA-ה אנזימים המבוססים עלEFR.
Approximate Labeled Subtree Homeomorphism

 shloup two mini时限 תושיהedef the sequence. (ALSHP) is a new model for the enumeration of labeled subgraphs of a given graph. The problem is to find all subgraphs that are isomorphic to a given graph, up to a certain tolerance. The algorithm is based on a novel approach to the problem of approximate homeomorphism.

The algorithm works by constructing a tree of possible isomorphisms, starting from a root node representing the original graph. The tree is then traversed using a depth-first search, with each node representing a subgraph of the original graph. The algorithm uses a combination of hashing and pruning to efficiently search the tree.

The algorithm has several advantages over previous approaches. It is significantly faster, and can handle much larger graphs. It is also more accurate, with a higher success rate in finding isomorphisms.

The algorithm has been implemented in C++, and is available for download from the authors' website. The software is open-source, and is licensed under the GNU General Public License.

The authors have tested the algorithm on a number of large graphs, including those arising in bioinformatics and other fields. The results show that the algorithm is able to find isomorphisms much more efficiently than previous approaches.

The paper concludes with a discussion of possible future work, including the extension of the algorithm to handle graphs with multiple labels, and the incorporation of additional constraints to improve the accuracy of the results.

The full paper is available for download from the authors' website.
החברה במרכיבים
הרשת
כיוון
הבעיות
בעיית
של
.cona
מי
מרכז
לא
יתבר
שה
הן
של

במיוחד
ראשית
דרישות
ביוכימיים
ניתן
לdataGridView
ב '@/דיבור
בibi
ושל
שני
קיים
לדוגמא
סיבוכיות
–
معنى
למצוא
אחרים
הנפוצה
שאילתות
התאמת
מאמצי
אקספוננציאלי
הוא
המייצג
כזה
יש
לקבוע
לMOST
 oluşu
בזירה
–
מטבוליים
וגרף
ההתאמה
או
נעשו
.לעשב
שה
ל.guna
ל掬
ל האי
זה
שונה
טבעיות
–
בבעיית
לא
מדויקת
ניהלה
לא
 рынке
ובividad
כמובן,
כ.Dock
והן
שונות
ליזון
ברד
>b
במקור
ביד
יד
lor
ל-msg
אפילו
לא
יודעים
שה
ל
ל_P
ל-
בתחום
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_SOL
ל-
ל dbg
ל_S
5.2

桶的Ini. $v_1$ 桶的桶的位置和衬垫在衬垫中的位置

5.3

5.4

5.5

5.6

5.7

A.1

A.2

A.3

A.4

A.5

A.6

A.7

A.8
3.1Tplahl hlokhty "העיצים". "העיצים"ddenby btohok bemyin hyokkotyi hlokhty "העיצים". "העיצים"
3.2Temah hlokhty "העיצים". "העיצים"
3.3Temah hlokhty "העיצים". "העיצים"
3.4Temah hlokhty "העיצים". "העיצים"
3.5Temah hlokhty "העיצים". "העיצים"
3.6Temah hlokhty "העיצים". "העיצים"
3.7Temah hlokhty "העיצים". "העיצים"

Approximated Labeled Subtree Homeomorphism

1. Introduction
2. Approximating Labeled Subtree Homeomorphism
3. Approximation to Genome Mapping
4. Subtree Homeomorphism in Metabolism
5. Conclusion and Future Work
המחקר נעשה בהנחיית פרופ’ שרון פינמר בפקולטה לפיזיקה אופטית והמכש.
אני מודה לפקודוני על התמיכה והחפעת נרחבת בשטח המחקר.
אלגוריתמים להמתת גרפים מทรניים

עם יישומים ביולוגיה מערבת

היבר על מחקר

לוש מילוי חלקית של תדרישות לפקולטה החיה
ודוקטור לפילוסופיה

אואלג רוכלניקו

הוגש לסניף הטכנין – מרכז טכנולוגיה לישראל
2007 דצמבר יוה"ח

כסלו תשמ"ח
אלגוריתמים להשתתף בקשיים מתרגישים
עם יישומיי לבולגוריה מיצירתית

אולג רוכלבך