Multiobjective Optimization for the Alignment of Protein Networks

by

Connor Clark

B.A., Grove City College, 2011

A thesis submitted to the Graduate Faculty of the University of Colorado at Colorado Springs in partial fulfillment of the requirements for the degree of Master of Science

Department of Computer Science

2014
This thesis for Master of Science degree by

Connor Clark

has been approved for the

Department of Computer Science

by

__________________________
Dr. Jugal Kalita, Chair

__________________________
Dr. Lisa Hines

__________________________
Dr. Rory Lewis

__________________________
Date
This thesis studies protein-protein interaction (PPI) network alignment, a young field that has attracted significant research interest recently. The goal of PPI network alignment is to approximately solve the subgraph isomorphism problem between the protein networks of two species, subject to biological constraints. It is hoped that network alignment can help in identifying orthologous proteins better than sequence similarity methods alone, as well as reveal shared pathways between species. We present the most comprehensive investigation of the relative performance of PPI network aligners performed to date, and analyze the alignments they produce. This reveals a distinct tradeoff between the first objective of maximizing the number of conserved edges between the two networks, and the second objective of maximizing the sequence similarity of aligned pairs of proteins. Leveraging these insights, we then develop a novel network aligner named Optnetalign (Optimizing Network Aligner) that significantly outperforms existing aligners. Optnetalign uses a multiobjective formulation of the problem that allows it to produce more diverse candidate alignments than using all other alignment algorithms combined. The state-of-the-art results obtained with Optnetalign better reveal existing deficiencies in network alignment research and indicate many possible directions for future work.
Dedicated to the clowder
Acknowledgments

We thank Jugal Kalita for supervising this work and providing immensely helpful advice. In addition, the comprehensive benchmarks presented in this thesis would have been impossible without the help of many network alignment researchers. We thank Gunnar Klau and Mohammed El-Kebir for their assistance with using the aligner NATALIE, Robert Patro for insightful discussion and help using the aligner GHOST, Shahriar Arab for his help in using the aligner NETAL, Rohit Singh for help using IsoRank, Cheng-Yu Ma for help in using PISwap, and Vikram Saraph for help in using MAGNA. Lastly, we thank the BioFrontiers Institute of the University of Colorado for funding portions of this work. Note: Chapters 2 and 3 originally appeared as: Connor Clark and Jugal Kalita, A comparison of algorithms for the pairwise alignment of biological networks, *Bioinformatics* 2014, 30(16):2351-2359. This work is reprinted here with permission of Oxford University Press.
# TABLE OF CONTENTS

1 Introduction ....................................................... 1  
   1.1 Objective ...................................................... 2  
   1.2 Outline of Thesis ............................................. 3  

2 Background ........................................................ 5  
   2.1 Introduction ..................................................... 5  
   2.2 Protein-Protein Interaction Networks and The Goal of Alignment ........ 7  
   2.3 Pairwise Global Alignment .................................... 8  
      2.3.1 Topological Similarity vs. Domain-Specific Similarity Measures ... 10  
   2.4 Pairwise Local Alignment ..................................... 11  
   2.5 Evaluating Global Alignment Quality .......................... 13  
   2.6 Existing Alignment Algorithms ................................. 16  
      2.6.1 Two-stage Aligners vs. Search-based Aligners .......... 22  

3 Benchmarks of Existing Aligners ............................... 24  
   3.1 Introduction and Methodology .................................. 24  
      3.1.1 Test Data .................................................. 24  
      3.1.2 Alignment Algorithms Evaluated .......................... 27  
   3.2 Results and Discussion ........................................ 30  
      3.2.1 Synthetic Test Results ..................................... 30  
      3.2.2 Real-World Test Results ................................... 34  
      3.2.3 Comparing Overall Performance ............................ 38  
   3.3 Benchmarking Conclusions ..................................... 42
5.4.3 Comparing Objectives ........................................... 74
5.5 Extracting Biological Data from Alignments: An Example and Difficulties . 81
5.6 Multiobjective Optimization Conclusions ................................. 83

6 Future Work and Conclusion ........................................... 84
6.1 Better Evaluation Metrics ............................................. 84
6.2 Similarity Estimation Functions ..................................... 85
6.3 Combining and Summarizing Alignments ............................. 87
6.4 Large-scale Network Analysis ....................................... 90
6.5 Conclusion .................................................................. 95

References ........................................................................ 96
TABLES

3.1 Numbers of edges in NAPAbench networks ............................. 26

3.2 Number of nodes and edges in Isobase networks ....................... 26

3.3 Parameter settings used in aligner evaluation .......................... 29

3.4 Typical running times of evaluated aligners ............................ 30

5.1 Correlation matrix for various objectives on the *S. cerevisiae* to *H. sapiens* alignment problem. ................................................. 76
FIGURES

3.1 $ICS$ of existing aligners – NAPAbench synthetic dataset. .................. 31

3.2 $LCSC$ size of existing aligners – NAPAbench synthetic dataset. ........... 32

3.3 Percentage of nodes aligned correctly for existing aligners – NAPAbench dataset. ......................................................... 33

3.4 $ICS$ of existing aligners – IsoBase dataset. ................................. 35

3.5 $LCSC$ size for existing aligners – IsoBase dataset. ......................... 36

3.6 $GOC$ for existing aligners – IsoBase dataset. .............................. 37

3.7 Existing aligners: correctly aligned nodes vs. $ICS$ – NAPAbench dataset. . 39

3.8 $GOC$ vs. $ICS$ for existing aligners – IsoBase dataset. ................. 40

3.9 $ICS$ vs. GO tradeoff for $C. elegans - D. melanogaster$ ................. 41

4.1 Weka plot of eigenvector centrality (x-axis) vs. authorities metric (y-axis) for aligned pairs in our alignment dataset. ...................... 50

4.2 Weka plot of PageRank (x-axis) vs. assortativity (y-axis) for our alignment dataset. ................................................................. 51

4.3 Results of Weka’s chi-squared attribute ranker. Assortativity is the most useful attribute by far, followed by closeness and eigenvector centrality. . 52
5.1 $S^3$ scores for the first experiment. ........................................ 70
5.2 GOC scores for the first experiment. ........................................ 71
5.3 Pareto fronts found by various aligners. ................................. 72
5.4 $S^3$ scores for the second experiment. .................................... 74
5.5 GOC scores for the second experiment. .................................... 75
5.6 Scatter plot of ICS scores vs. GOC scores on the $S. cerevisiae$ to $H. sapiens$ alignment problem. ....................................................... 77
5.7 Scatter plot of EC vs. ICS on the $S. cerevisiae$ to $H. sapiens$ alignment problem. ................................................................. 78
5.8 Scatter plot of GOC scores vs. sum of bit scores on the $S. cerevisiae$ to $H. sapiens$ alignment problem. ....................................................... 79
5.9 Scatter plot of $S^3$ scores vs. GOC scores on the $S. cerevisiae$ to $H. sapiens$ alignment problem. ....................................................... 80
6.1 Problems with the largest common component evaluation metric. ... 86
6.2 Schematic of a map-reduce algorithm for large-scale multiple alignment . 94
CHAPTER 1

INTRODUCTION

Protein-protein interaction (PPI) networks are graphical representations of a species’ proteome. The nodes of such a network are individual proteins, and two nodes are linked by an edge if they interact to perform some biological function. Recent experimental techniques have allowed large databases of protein interactions to be created, and researchers are now faced with interpreting large amounts of this network data.

One way of analyzing these networks is by overlaying two or more of them on top of each other to find their maximal overlap. This overlapping region would hint at the PPI network of the two species’ common ancestor, and allow us to identify proteins of similar function between the two species. When one of the species is better studied than the other, this could allow us to transfer knowledge of one species’ proteome to the other’s. This process is called network alignment, in analogy to the process of aligning genetic sequences.

In the past few years, there has been a flurry of research activity on this problem. The youth of this field is problematic, however. Most of the notable alignment algorithms were developed by separate groups concurrently, and were published in such fast succession that they were not rigorously compared to one another. Even the specifics of what network alignment meant differed slightly. Most aligners required that the resulting alignment match every node of the smaller graph to the larger, while others omitted nodes for which no good match could be found. Furthermore, these separate groups all developed similar, but non-identical, ideas on how to evaluate the results of these alignment algorithms. Some focused on maximizing the topological overlap of the two networks, leaving biological similarity of
matched proteins as a secondary objective, while others did the opposite. Even given the agreement on these two overarching objectives, the specific methods used to measure them differed, and all of them operated under the assumption that the two goals of topological fit and biological fit did not conflict.

The recent flurry of activity in network alignment has left us with a great many aligners, all of very different design and with subtle differences in their stated goals. This directly informs the objective of this thesis.

1.1 Objective

The objective of this thesis is to make sense of the overall network alignment research area, and then to improve upon the existing state-of-the-art. We do this by performing the first large-scale comparison of many aligners, allowing us to better understand the large variety in aligner design. We discover that the tradeoff between topological fit and biological fit is much more dramatic than previously thought. Since existing aligners all explore small, non-overlapping portions of this objective space, we introduce a new way of performing alignment, based on the field of multiobjective optimization, that not only outperforms existing aligners, but also explores a very wide region of potential tradeoffs between the two alignment objectives. We hope that our results change the way the alignment problem is framed, and inspire a new generation of research that takes into account that, for a given pair of PPI networks, hundreds of potential alignments can be created that outperform existing aligners, and these alignments often disagree greatly in the pairs of nodes they align.
1.2 Outline of Thesis

This thesis is organized into six chapters. In the second chapter, we lay out the problem of network alignment in detail, describing the different measures of topological and biological fit that have been introduced and the networks we use for evaluation, and providing a literature review of existing alignment algorithms. We also present a scheme for classifying these aligners into a few broad categories, depending on their design.

In the third chapter, we benchmark many existing aligners, revealing that the variety and diversity of possible alignments for a given pair of networks is very high. We show that the overall performance of existing aligners varies widely—some dramatically outperform others, and most skew strongly towards maximizing one objective while doing poorly on the other.

In the fourth chapter, we cover our preliminary investigation into using machine learning to try to create alignments. Using synthetic PPI networks, generated by models of PPI network evolution, we attempt to train a classifier to identify orthologous protein pairs when given standard graph-theoretic measures of the subgraph around each node as input. Despite the fact that this approach is largely unsuccessful, it provides interesting insights into the problem itself, since other aligners have successfully used node similarity functions based on topological data. We include these results to help inform future work, and to better introduce our work in the next chapter, which eschews the common practice of estimating node similarity entirely.

In the fifth chapter, we present a multiobjective metaheuristic algorithm to perform PPI network alignment. Our experiments on using swap-based hill climbing lead us to create a custom-built hybrid metaheuristic that primarily uses hill climbing, augmented by crossover and mutation, to produce results that dramatically outperform most existing
aligners. Our approach generates an enormous number of candidate alignments for each pair of networks, demonstrating that a huge number of high-quality alignments can be created for a given pair of networks, and that these alignments often agree very little on which proteins should be matched.

These results suggest new lines of investigation. In the sixth chapter, we review what we have discovered, and outline what this suggests about future work. We believe that our findings must change the way global PPI network alignment is framed, and further investigation must transition away from building yet more alignment algorithms, and instead towards understanding and interpreting alignments that have been produced by these algorithms.
CHAPTER 2
BACKGROUND

In this chapter, we introduce protein-protein interaction networks, define the problem of global PPI network alignment precisely, overview how alignments are evaluated, and provide a literature review of existing alignment algorithms.

2.1 Introduction

Many areas of bioinformatics now use and produce network data, including protein-protein interaction (PPI) networks and gene coexpression networks, and the tools and techniques of graph theory are being brought to bear to produce new techniques for biological network analysis. Comparing two biological networks is a particularly challenging problem, since many interesting questions we might ask of these networks are computationally intractable to answer [6]. Here, we focus on the task of aligning two PPI networks from different species. That is, we want to find a mapping from the nodes of one network to the nodes of another, in such a way as to maximize the topological and biological similarity of the pairs of nodes which are aligned to one another. This allows for the identification of both homologous proteins as well as similar modules or pathways in the networks themselves. Successes of PPI network alignment so far include uncovering large shared subnetworks between species as diverse as S. cerevisiae and H. sapiens, and reconstructing phylogenetic relationships between species based solely on the amount of overlap discovered between their PPI networks [57, 58]. Most papers in the literature report promising results in cre-
ating alignments that do indeed show large regions of biological and topological similarity between the PPI networks of various species.

The ultimate goal of network alignment is to transfer knowledge of protein function from one species to another. Since sequence similarity metrics such as BLAST bit scores [4] are not conclusive evidence of similar function, the purpose of aligning two PPI networks is to supplement sequence similarity with topological information so as to identify orthologs as accurately as possible. The primary challenge in designing such an aligner is to accurately estimate the topological similarity of two nodes, and to combine that with sequence similarity to produce an alignment. The aligners published so far vary widely in their approaches to doing so, and some aligners are much better at optimizing for one of these goals than the other.

An exact solution to the network alignment problem is unattainable for practical problem instances. The problem of global alignment is equivalent to the subgraph isomorphism problem, which is NP-Complete [18], so aligners settle for approximate solutions. The variety of approaches in use and the lack of a popular standard solution present a difficult situation for those who would simply like to compare some biological networks that they have produced in the course of their research. These algorithms differ greatly in the quality of the alignments they produce, and even more greatly in their compute time.

In this thesis, we focus on pairwise and global alignment algorithms. Pairwise alignment algorithms align two graphs only. They are contrasted with multiple alignment algorithms, which try to find transitive alignments between more than two input networks at a time [29,42,43,61,82,87]. Borrowing terminology from sequence alignment, we also distinguish between global alignment algorithms, which attempt to find a single overall alignment from one network to another, and local aligners, which may output several mutually in-
compatible alignments for the input networks [9, 28, 44, 53, 60, 86]. Local alignment is more useful when we desire to identify several potential orthologs per input protein, while global alignment is more helpful for identifying larger conserved networks that are indicative of a common ancestor. Global alignments can also be somewhat easier to interpret, since the produced mapping is one-to-one. Since pairwise global aligners have been much more popular in the recent literature, and because it is unclear how to compare a global algorithm to a local algorithm, or a pairwise aligner to a multiple aligner, we focus on pairwise global aligners in this thesis.

2.2 Protein-Protein Interaction Networks and The Goal of Alignment

A protein-protein interaction (PPI) network is a graphical representation of all the proteins in a given organism, where the nodes of the network represent proteins, and a link between two proteins indicates that they interact to perform some biological function. Protein interaction is measured through a number of different experimental methods, with the most common being coimmunoprecipitation [1]. A large amount of effort has gone into collecting and curating the results of many protein interaction experiments, and the results are stored in databases such as DIP [94], ISOBase [72], BIOGrid [13], and STRING [31]. A related research effort in this area is the Gene Ontology (GO), a large database annotating genes and proteins with information about their function, the biological processes in which they participate, and the cellular components in which they are found [5]. The GO database is constantly being updated and revised as new discoveries are made. An important area of research is the transfer of GO annotations from proteins in one species to orthologous proteins in another. Some species have more complete GO annotation data than others, so
it is very desirable to find a way to accurately and automatically predict GO annotations for unannotated proteins in less-studied species.

One attempt at such annotation transfer is presented in a recent article by Milenkovic et al. [67], which focuses on transferring information on aging-related proteins from well-studied model species such as *C. elegans* to human proteins. Since long human lifespans make aging-related studies much more difficult, it is easier to study aging in other species, identify aging-related proteins in those species, and then transfer that information to human proteins by performing PPI network alignment. The study shows that network alignment can produce useful predictions of involvement in aging in previously unannotated human proteins [67].

In short, PPI network alignment is a way to use both the sequence similarity between proteins, as well as the topology of PPI networks, to find likely orthologous proteins between species. The goal is to create algorithms that can automatically find likely subnetworks of two species’ PPI networks that are common to the two species, and from there form hypotheses about protein function in species for which less information exists. A single run of an alignment program can generate thousands of candidate ortholog pairs.

### 2.3 Pairwise Global Alignment

Of the many proposed methods for analyzing biological networks, global network alignment is one of the most computationally ambitious. We are given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, whose vertices represent proteins, and the presence of the edge $(u, v)$ in $E_1$ (or $E_2$) indicates that the two proteins represented by $u$ and $v$ interact in $G_1$ (or $G_2$). Most aligners assume, without loss of generality, that $|V_1| < |V_2|$.
The problem of pairwise alignment is to find a one-to-one function $f : V_1 \to V_2$ that maps each node in $V_1$ to the node in $V_2$ that it best matches \footnote{As a shorthand to make several equation more readable, we will treat $f$ as a function on edges, as well. In this case, $f((u, v))$ is simply a more readable shorthand for $(f(u), f(v))$.}. It must be noted that some algorithms produce a partial function, abstaining from mapping nodes that cannot be matched well.

Most aligners decompose the process of producing a matching into two steps. First, for each pair of nodes in $V_1 \times V_2$, we compute their similarity, by examining the local topology of the graph around those two nodes, and by their sequence similarity, as measured by BLAST bit scores or E-values. Second, taking the similarities between these nodes as weighted edges in a bipartite graph with two sets of nodes $V_1, V_2$, we solve the maximum weight bipartite matching problem to generate a mapping from $V_1$ to $V_2$. The pairwise alignment software available differs primarily in how they handle these two steps, with most innovation being focused on the first step of estimating the topological similarity of two nodes. This process is just a general schematic that applies to most aligners, which we describe as two-stage aligners in our more detailed discussion below. Some, such as NATALIE 2.0, do not follow the two-step process, and instead optimize a relaxed version of the problem directly.

We must stress that all existing aligners only approximately solve this problem, and they generally introduce approximations in two ways. First, they introduce a relaxed problem definition. Second, they use a heuristic algorithm to approximately solve the relaxed problem. For instance, GRAAL frames the problem as matching nodes to one another in such a way as to maximize their graphlet signature similarity (a measure of how many small subgraphs of various shapes are in both nodes’ respective neighborhoods), in the hopes that maximizing this metric between all aligned pairs of nodes will produce a biologically-
informative alignment [57]. Then, a greedy matching heuristic is used that aligns the most similar pair of nodes first, and then works outward, aligning their neighborhoods. When evaluating this algorithm, it is not clear how much of its performance is attributable to the metric of graphlet signature similarity, and how much is due to using a matching algorithm that prefers to map neighbors to neighbors. All we can do is evaluate how well the heuristic solution to the relaxed problem solves our original problem. If a given aligner performs poorly, it could be that the relaxed problem is a good choice, but that choice is hamstrung by a poorly-designed heuristic to solve it. Without being able to swap out the parts of these aligners, however, all we can do is evaluate their final results. One study mixed-and-matched the stages of two aligners [67], but no further work has been done in this area.

2.3.1 Topological Similarity vs. Domain-Specific Similarity Measures

Broadly speaking, there are two ways to estimate the similarity of two nodes. One may assess the shape of the network around that node through a variety of metrics such as degree, eccentricity, betweenness, or the more recently developed graphlet degree [78]. Then, nodes that appear to be in topologically similar regions of their respective networks are considered likely matches. Sequence similarity information is also very useful, and BLAST bit scores or E-values have been popular measures of node similarity. In most alignment algorithms, both topological information and sequence similarity information are needed, though there are a few notable exceptions described below.

Both topological similarity techniques and sequence similarity have advantages and disadvantages. It has been argued that over-reliance on topological similarity can be misleading, since actual complexes may appear disconnected in current noisy, incomplete
datasets, and so sequence similarity information is essential to produce the best alignment possible [39]. Sequence similarity scores also have their problems, since the actual level of sequence similarity between two proteins that serve a similar function can vary [14]. Previous work has suggested [74] that existing aligners uncover a tradeoff between topological fit and biological fit when constructing alignments— it is not possible to jointly maximize both. Our work in Chapter 3 presents much further evidence of this. This tradeoff can be quite dramatic. For example, on the *C. elegans* - *D. Melanogaster* alignment problem, NETAL [69] achieves an $S^3$ of 0.41 and a GOC of 12, while PINALOG [77] manages a meager $S^3$ of 0.065 and an impressive GOC of 230. When existing aligners are all plotted for one of these problems, we see a distinctive curve of tradeoffs between these two objectives. We also find that these aligners produce alignments in non-overlapping regions of the objective space, even for those aligners that provide a parameter for controlling the tradeoff between the two objectives. As we will show in Chapter 5 our metaheuristic aligner is the only aligner that was designed to explicitly explore the possible tradeoffs between these two objectives, and to output many alignments for further analysis.

After an alignment is constructed, alignment papers use more computationally-intensive methods to ensure that they have produced alignments of good topological and biological fit, which we will discuss below.

### 2.4 Pairwise Local Alignment

While we focus on global alignment in this thesis, it is important to contrast it with local network alignment. Local network alignment, like local sequence alignment, tries to find many small regions of local similarity that may be mutually incompatible [9,28,44,60]. These smaller similar regions are often called structural motifs [9]. Formally, we can think
of local alignment as attempting to find a function $f : V_1 \rightarrow \mathcal{P}(V_2)$, which maps the nodes of $G_1$ to some small number of nodes $G_2$ to which they are most similar. For example, a single node $v$ in $V_1$ might be mapped to three highly similar nodes $w,x,y$ in $V_2$. The drawback of this approach, which contributed to it falling out of favor compared to global alignment, is that the results are more difficult to interpret. Consider the case where two neighboring nodes $x,y$ in $V_1$ are each mapped to several nodes in $V_2$. The nodes $x$ and $y$ may be mapped to several non-neighboring nodes in distant parts of $G_2$. If we want to reconstruct what common structures $G_1$ and $G_2$ share, we must consider every combination of the potential matches we are given for those two nodes. When 3,000 nodes have been aligned to an average of, say, 5 nodes each, interpreting the results may be as difficult a problem as the original alignment! Local alignment also became less popular after it was discovered that very large regions of apparent structural similarity can be found by global alignment [58]. Given that discovery, finding small local regions of similarity became less popular.

Local alignment does have a conceptual advantage over global alignment, however. Several popular models of protein network evolution propose that new proteins evolve through gene duplication [73, 93]. Once a gene has duplicated, its duplicate may mutate more freely, eventually producing some novel protein, while the original version continues to produce the original protein that is often necessary for the organism’s survival. Such models of evolution suggest that local alignment’s identification of multiple potential orthologs is truer to the underlying evolutionary dynamics we are trying to elucidate. This is reinforced by observations both in this thesis and in other papers [16, 74] that global alignments produced by different aligners may both appear to be of high quality while sharing as few as 5% of their aligned pairs. A tool that produces a single global alignment may be arbitrarily
inflating the significance of aligning one node to another, when several other nodes could
be equally good matches.

Local alignment evaluation never reached the level of uniformity and standardization
found in the global alignment literature, and, since local aligners state their goals somewhat
differently, little comparison between methods has been made. Generally, these aligners are
evaluated in terms of the number of GO terms shared by aligned nodes, with much less
emphasis placed on topological consistency of the alignment.

2.5 Evaluating Global Alignment Quality

We now review methods for evaluating the quality of a global pairwise aligner’s output.
Several have been proposed. When evaluating with real biological data, the true alignment
between two PPI networks is unknown, so one cannot simply report the percentage of
nodes mapped to their true orthologs, as we do with synthetic data. For pairwise alignment
algorithms, the metric of edge correctness ($EC$), proposed by Kuchaiev et al. [57], reports
the percentage of edges in $G_1$ that are conserved under the produced mapping to $G_2$:

$$EC = \frac{|f(E_1) \cap E_2|}{|E_1|}$$  \hspace{1cm} (2.1)

where $f$ is the mapping produced by the alignment algorithm. Edge correctness is suggestive
of a good alignment, but since it is always possible that two biologically unrelated edges
have been mapped to each other, even 100% edge correctness is not conclusive evidence of
a correct alignment.

While edge correctness is an intuitive measure of alignment quality, a more nuanced
alternative that has been proposed recently [74] is the induced conserved structure (ICS)
score. This extends EC with a further intuition: if some region of $G_2$ is dense, then a sparse
region of $G_1$ could be mapped to it in many different ways. We would rather align a sparse region of $G_1$ to a sparse region of $G_2$, to increase our confidence that the alignment is not merely a coincidence. The ICS score penalizes alignments that map to denser subgraphs of $G_2$. Let $G[V]$ denote the induced subgraph of $G$ on the vertices $V$. Then the induced conserved structure score of an alignment $f$ from $G_1$ to $G_2$ is [74]:

$$ICS = \frac{|f(E_1) \cap E_2|}{|E_{G_2[f(V_1)]}|}. \quad (2.2)$$

ICS tends to give us a slightly lower score than EC with existing alignment algorithms. Since this score is more conservative, we use it for our benchmarks of existing algorithms in Chapter 3. However, with the metaheuristic aligners we develop in Chapter 5, we discovered that aligners can “cheat” on ICS by minimizing the denominator greatly while only increasing the numerator slightly. This gives large ICS scores, even though very few of the edges in $E_1$ have been conserved. For this reason, we also make use of the $S^3$ metric, which is a tweak to ICS first presented by Saraph and Milenkovic [83].

$$S^3 = \frac{|f(E_1) \cap E_2|}{|E_1| + |E_{G_2[f(V_1)]}| - |f(E_1) \cap E_2|}. \quad (2.3)$$

In addition to penalizing dense-to-sparse mappings like ICS, $S^3$ also penalizes sparse-to-dense mappings. Not only does this close the loophole in ICS that our metaheuristic algorithms exploit, but also gives an even stricter metric for evaluating the topological fit of an alignment.

Another popular metric of alignment correctness is the size of the largest connected component shared by the two graphs that is found by the alignment. This largest connected shared component (LCSC) is the largest connected subgraph of $G_1$ that was found to also exist in $G_2$. A larger LCSC implies that we have found a larger amount of shared structure
between the two PPI networks. Though it has been noted that current network data is woefully incomplete [39], our results below show there appears to be some relationship between larger LCSC and the number of correctly mapped nodes in synthetic networks. LCSC is also called “largest common connected subgraph” (LCCS) in some sources, and we use the two terms interchangeably in this thesis.

Measures of agreement derived from biological information are also popular. All the literature on PPI network alignment makes use of protein annotations from the Gene Ontology (GO) Database [5] to evaluate the accuracy of their alignments, by comparing the similarity of GO annotations between aligned proteins. Since most network alignment papers so far present entirely novel methods, most of the emphasis on evaluating the biological quality of an alignment has been focused on verifying that the alignment is plausible, by checking whether aligned nodes are biologically similar in terms of the Gene Ontology. Much less work has been done so far on deciding which annotations seen in one protein should be predicted as being present in the protein to which it has been aligned. A given protein is mapped to any number of GO terms, and even true orthologs may not have the same set of GO terms, both due to incompleteness of the GO data as well as divergence of protein function.

Several ways of measuring GO term agreement have been used in the literature. The simplest, and most common, is to merely count the number of aligned pairs with more than \( n \) GO terms in common [56]. A more sophisticated method, that requires statistical information on GO term frequency that is not always readily available, is to use a measure of semantic similarity such as Resnik’s semantic similarity method [74]. In our own evaluations, we take the middle ground, using a simpler method that measures the overlap of GO terms between aligned nodes. Aladag and Erten [3] define \textit{GO consistency} \((GOC)\) as:
\[ GOC(u, v) = \frac{|GO(u) \cap GO(v)|}{|GO(u) \cup GO(v)|} \]

for an aligned pair of nodes \( u \in V_1 \) and \( v \in V_2 \), where \( GO(u) \) denotes the GO terms associated with the protein \( u \) that are distance 5 from the root of the GO hierarchy. Limiting the set of terms in this way prevents score inflation caused by counting both more and less specific GO terms. Since network aligners often use sequence similarity to create an alignment, we also experiment with restricting this score to only use GO terms with experimental evidence codes, which excludes GO terms that have been assigned on the basis of sequence similarity. We then report the sum of the GO consistency over all alignment pairs for each alignment produced.

2.6 Existing Alignment Algorithms

As noted above, many alignment algorithms have been released in the last few years. These tend to have few design features in common, and approach the problem in different ways. As we will see in Chapter 3, they also vary wildly in their performance on the problem. In this section, we review the most notable alignment algorithms published since the first pairwise global aligner was published in 2008.

While virtually every alignment algorithm published since is claimed to perform better, IsoRank [88] was the first and most widely-cited algorithm used for global network alignment, and it remains a popular baseline for measuring the performance of new algorithms. With IsoRank, topological similarity between two nodes is defined recursively—two nodes are similar if their neighbors are similar. This intuition is formalized as an eigenvalue.

\(^2\)If a protein has no GO terms at distance 5 from the hierarchy, it will not contribute to \( GOC \). There are a large number of proteins with many terms at distance 5, however, so we still get very informative scores.
problem, and the similarity score matrix is iteratively refined using the power method for computing an eigenvalue. This is combined with sequence data to find a mapping using a seed-and-extend algorithm.

GRAAL [57] is the original algorithm in the GRAAL series. It is notable for being the first algorithm to use topological data exclusively to construct an alignment. GRAAL determines topological similarity by counting “graphlets” – small induced subgraphs. For each node in the input networks, GRAAL computes how many times the node occurs in each graphlet. These counts are used to construct a graphlet degree signature, and the distance between the graphlet degree signature of two nodes is used as a measure of topological similarity between the nodes of the networks being aligned. A heuristic matching algorithm is used that first aligns the two nodes that are most similar, then works outward to align their neighbors, until all nodes in $V_1$ have been aligned. As MI-GRAAL is recommended by GRAAL’s authors as a superior solution, this algorithm is included in our benchmarks in Chapter 3 largely because of its historical impact on the network alignment literature.

MI-GRAAL [58], from the family of GRAAL algorithms, combines several different measures of topological similarity along with sequence similarity. The user can decide which topological measures to use in a given alignment. In our benchmarks, we use the combination of topological similarity measures that is reported to work best: graphlet degree signature distance, degree difference, and clustering coefficient difference. When considering whether to align a given pair of nodes, each of these similarity measures is treated as a separate vote for or against aligning the two nodes. MI-GRAAL then uses a seed-and-extend approach to greedily build up an alignment by using the Hungarian algorithm on successive neighborhoods of already-aligned nodes. Notably, MI-GRAAL was one of the first aligners for which sequence similarity data was optional, and even when using only
topological information, MI-GRAAL is reportedly able to find alignments accurate enough to reconstruct phylogenetic trees from given PPI networks [58].

C-GRAAL [64] is a more recent algorithm in the GRAAL family. It differs from MI-GRAAL in that it uses only graphlet degree signature and sequence similarity to construct its alignments, and it uses a different heuristic matching algorithm. The matching algorithm works by conducting repeated seed-and-extend iterations that first align the most similar pair and then aligns neighbors of already-matched nodes. When no more unmatched neighbors exist, it finds a new seed from the most similar pair that is still unaligned and repeats the process, until all nodes are aligned. This neighbors-based approach helps to maximize the edge correctness of the alignments produced. It is reported that C-GRAAL obtains lower edge correctness scores than MI-GRAAL, but performs better with respect to conserved functional orthology between aligned nodes and finds larger connected shared components [64].

NATALIE 2.0 is a pairwise aligner that uses both topological and sequence similarity information for alignment [27]. Unlike most aligners, which proceed in a two-stage fashion as described above, NATALIE 2.0 formalizes the alignment problem as a Lagrangian relaxation approach to solving an adaptation of an integer linear programming problem, which attempts to optimize topological and sequence similarity of aligned nodes. NATALIE also differs from the other aligners here in that it frames the problem as finding a partial function $f : V_1 \rightarrow V_2$, which allows NATALIE to leave some nodes unaligned. We found that NATALIE puts this difference to good use, abstaining from aligning nodes for which it cannot find a good alignment. NATALIE also restricts its alignment by incorporating a user-configurable cutoff for sequence similarity, below which it will not consider mapping two nodes together. This improves the execution speed considerably by allowing the use
of the successive shortest paths variant of the Hungarian algorithm, which has better time
complexity than the standard version.

GHOST [74] calculates spectral signatures from the Laplacian of the subgraphs around
each node of the input graphs. These signatures can be saved for each graph and reused in
further alignment runs, which helps to save computation time. The similarity between two
nodes is then defined as the distance between the spectral signatures of the nodes. Since
these are matrix operations, parallel algorithms are used to decrease GHOST’s running
time. For the matching stage, GHOST uses a seed-and-extend approach, where the most
similar pair of nodes is aligned as the first seed, and then the neighborhood around these
nodes is mapped by solving the spectral relaxation of a quadratic assignment problem, which
assigns a match confidence to each pair in the neighborhood. The highest-confidence match
found is aligned and used as the seed for the next iteration. On real-world PPI networks, it
is reported that GHOST finds a larger connected common subgraph than MI-GRAAL and
does significantly better at matching nodes with shared GO annotations [74].

SPINAL [3] is a rather recent pairwise alignment algorithm that claims much better
performance in terms of memory usage, speed, and accuracy over MI-GRAAL. It uses a
two-pass matching algorithm consisting of “coarse-grained” and “fine-grained” steps. The
coarse-grained step iteratively improves a matrix $P$ of estimated match confidence for each
pair of nodes by taking into account the confidence of matching their neighbors that was
computed in the previous iteration. After $P$ has converged, the fine-grained stage begins,
which uses a seed-and-extend algorithm to construct the alignment. Additionally, on each
iteration of the seed-and-extend process, local search is performed to increase the number
of conserved interactions directly. The authors report significantly better performance than
MI-GRAAL in both runtime and alignment quality. Furthermore, SPINAL has two dis-
tinctive modes, 1 and 2, with mode 1 performing the coarse-grained phase and then simply performing a maximum-weight bipartite matching, while mode 2 performs both the coarse- and fine-grained stages.

NETAL [69] is another very recent pairwise algorithm that boasts great improvements in execution speed over older algorithms such as MI-LGRAAL. Its performance on noisy data is reportedly improved over MI-LGRAAL as well, aligning nearly three times as many nodes correctly on a network with 5% noise in its edge set [69]. NETAL is also notable for being available for use online. NETAL works by recursively defining topological similarity in a manner similar to IsoRank, where nodes are similar if their neighbors are similar. Then, given this topological similarity matrix, it refines an interaction score matrix by estimating how many interactions will be conserved if the given nodes are aligned. The alignment is constructed greedily by mapping the nodes with the best interaction score together, and after each pair of nodes is aligned, the interaction score matrix is updated for selecting the next pair to align. The current version of NETAL uses topological similarity only, but the paper in which it is presented states that a version that uses sequence similarity is forthcoming [69].

PINALOG [77] is a pairwise aligner that identifies dense subgraphs, called communities, within the input networks to find regions of similarity between the two networks being aligned. It first finds a mapping from the communities in one graph to the communities in the other, and then, for each mapped pair of communities, matches the nodes within them. The community information is the only topological information incorporated. The similarity of two communities is determined by assessing the sum of the sequence similarity of the best alignment created between those two communities by matching their constituent nodes

[^3]: http://bioinf.modares.ac.ir/software/netal/
by sequence similarity. Like NETAL, PINALOG is available both through a web interface
and as a stand-alone executable. PINALOG is reported to find alignments with edge cor-
rectness similar to MI-GRAAL, but the alignments found tend to be better matches with
respect to functional orthology data [77]. Like most aligners, PINALOG uses both topolog-
ical and sequence similarity information in constructing its alignment, but it automatically
determines the tradeoff between the two.

MAGNA [83] is, prior to our work in Chapter 5, the only genetic algorithm aligner. It
represents alignments as permutations of the nodes of the larger network, and the first
$|V_1|$ nodes of that permutation are interpreted as being mapped to corresponding nodes of
$V_1$, while the rest are considered unaligned. MAGNA uses a novel crossover operator that
produces a new alignment from two parents by computing a permutation that is halfway
between the two parents in terms of swapping elements of their permutations. No mutation
operator is used, but MAGNA is still able to find good topological matches in a reasonable
amount of time. MAGNA was also used to try to further refine the alignments produced by
existing aligners, to great success. MAGNA considers only topological fit when evaluating
the fitness of the alignments it has produced, using the $S^3$ metric discussed above. Since
MAGNA is much more similar to our work, we evaluate it in Chapter 5 when we present
our own algorithm, instead of in Chapter 3 with the aligners enumerated above.

PISwap [14] can also be seen as a precursor to our work in Chapter 5. This aligner
first constructs an optimal matching between the nodes of the networks by bitscore, and
then attempts to refine the alignment by performing swaps that increase edge correctness
without decreasing the sum of the bitscores of aligned pairs. Like MAGNA, we wait until
Chapter 5 to evaluate PISwap, since it bears similarities to our metaheuristic aligners.

\footnote{http://www.sbg.bio.ic.ac.uk/~pinalog/}
2.6.1 Two-stage Aligners vs. Search-based Aligners

The aligners above can be categorized as belonging to one of two classes. The first class, which has been much more popular, is what we call two-stage alignment. Such aligners first compute some estimate of the similarity of each pair of nodes in $V_1 \times V_2$ using some topological similarity metric and (in most cases) BLAST bit scores or E-values, and then use this similarity matrix to create a maximum-weight bipartite matching between the nodes of $V_1$ and $V_2$. A simple strategy for this matching is the Hungarian algorithm, which produces an optimal matching [14,66]. However, because the similarity scores used are themselves only approximate, the $O(n^3)$ time complexity of the Hungarian algorithm is generally not worth the time, and most aligners favor faster, greedy matching algorithms. Many aligners, such as GRAAL [57], MI-GRAAL [58], and GHOST [74], use variants on a “seed-and-extend” method, where the most similar pair of nodes is aligned first, and then nodes neighboring that pair are matched.

The second, and much smaller class, that is sometimes used in network alignment are search-based aligners. These aligners use heuristic or metaheuristic algorithms to continuously refine a single alignment or a population of alignments. With such algorithms, the alignment objectives that these aligners are trying to optimize are computed continuously as the alignment is constructed, instead of being measured afterward as they are in the two-stage aligners. As far as we are aware, only a handful of search-based aligners exist. These are SIM-T [54], which uses tabu search, where “moves” are additions and removals of aligned pairs from the alignment; PISwap [14], which uses the 3-opt heuristic to interchange the assignments of three aligned pairs at each step; and MAGNA [83], which uses a genetic algorithm that attempts to maximize either $EC$, $ICS$, or $S^3$. SPINAL [3] also performs some amount of swap-based hillclimbing in its “fine-grained” matching stage.
Outside of the biological network alignment literature, the use of metaheuristic search for graph matching has been researched as well, but these approaches are less relevant as they only optimize topological fit and are applied to graphs much smaller than those seen in bioinformatics [7, 15, 20, 62].

The novel aligner we present in Chapter 5 falls into the search-based category, because we believe the two-stage paradigm has a number of severe weaknesses. First of all, two-stage aligners typically introduce both a novel way of computing pairwise node similarity along with a new matching algorithm. Little work exists on mixing-and-matching approaches to the two stages to discover whether the first or second stage is more responsible for the results of a given aligner. To our knowledge, the only published attempt at this considered only MI-GRAAL and IsoRankN [67]. Since these algorithms do not track alignment quality as they construct their alignments, it is very difficult to understand what portions of a two-stage algorithm are responsible for the quality of the results. This makes it difficult to create a new algorithm of this type. The second deficiency of the two-stage process, as Saraph and Milenkovic [83] recently observed, is that these two-stage aligners, while hoping to optimize metrics such as EC and GO term agreement, actually optimize ad-hoc estimated similarity scores that have no clearly established relation to standard alignment evaluation metrics. The existing literature provides little explanation of why or whether such similarity scores would be expected to work. In contrast, search-based aligners focus on maximizing the actual alignment objectives that are currently accepted in the alignment literature.
3.1 Introduction and Methodology

Given the rapid pace at which new alignment techniques are being published, many of the algorithms discussed in Chapter 2 have never been directly compared to one another. Therefore, we include extensive comparisons and benchmarks in this chapter. We make use of a recently developed framework for testing alignment algorithms with synthetic PPI network data from the NAPAbench aligner benchmark dataset [81] as well as a real-world PPI dataset, Isobase [72]. We find great differences in the quality of the alignments produced by existing alignment programs.

3.1.1 Test Data

To perform our evaluation of existing alignment algorithms, we make use of the NAPAbench synthetic PPI network data, which was created specifically for benchmarking network alignment algorithms [81]. This benchmark data is made using state-of-the-art algorithms for simulating the evolution of PPI networks, and can construct arbitrary phylogenetic trees with PPI networks of customizable sizes. With this synthetic data, the exact topology and orthology of the two networks are known. Since real orthology and PPI network data are constantly improving in terms of both completeness and accuracy, benchmarking on a perfect dataset gives us a good idea of the upper bound of each alignment algorithm’s performance, rendering our evaluation independent of the quality of biological data available at the time it is published. Furthermore, with the known true alignment
available as ground truth, we are able to get a much better idea of an alignment’s quality. NAPAbench has previously only been used to benchmark several older pairwise alignment algorithms and some more recent multiple alignment algorithms [82]. Here, we use it to test more and newer pairwise algorithms.

Because of the number of algorithms benchmarked, and the high time requirements of some, we use a subset of the standard NAPAbench dataset, consisting of nine pairwise alignment problems. This includes three problems from each of the three PPI network evolution models used to generate the synthetic network data. These are the duplication with random mutation (DMR) model [73], the duplication-mutation-complementation (DMC) model [93], and a crystal growth (CG) model that has recently been proposed [46]. Each of these alignment problems involved aligning a network of 3000 nodes to a network of 4000 nodes. However, each of the problems differed in the number of edges and network topology. Table 3.1 shows how many edges are present in each problem.

Additionally, we use experimentally-derived PPI network data from IsoBase [72], supplemented with sequence similarity and GO annotation data from the supplementary information file of [3]. The IsoBase dataset is very popular for evaluating alignment algorithms, so we make use of it to further ease comparison with alignment algorithms that may be published in the future. It is important to test alignment algorithms on real datasets, since such data is noisy and incomplete. How well an alignment algorithm handles the spurious and missing edges that occur in real data is an important part of understanding its performance. However, we don’t know the true alignment between empirically-derived PPI networks, so performance metrics for alignment on real datasets are inherently less informative. The number of nodes and edges in each of the Isobase networks, as well as the names of the species to which each network belongs, are given in Table 3.2.
### Alignment Experiment

<table>
<thead>
<tr>
<th>Network</th>
<th>$G_1$ Edges, $G_2$ Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crystal Growth 1</td>
<td>(11986, 15986)</td>
</tr>
<tr>
<td>Crystal Growth 2</td>
<td>(11987, 15987)</td>
</tr>
<tr>
<td>Crystal Growth 3</td>
<td>(11984, 15984)</td>
</tr>
<tr>
<td>Duplication-Mutation-Complementation 1</td>
<td>(6090, 8112)</td>
</tr>
<tr>
<td>Duplication-Mutation-Complementation 2</td>
<td>(6361, 8310)</td>
</tr>
<tr>
<td>Duplication-Mutation-Complementation 3</td>
<td>(6077, 8347)</td>
</tr>
<tr>
<td>Duplication with Random Mutation 1</td>
<td>(6017, 8238)</td>
</tr>
<tr>
<td>Duplication with Random Mutation 2</td>
<td>(5739, 7602)</td>
</tr>
<tr>
<td>Duplication with Random Mutation 3</td>
<td>(6457, 8431)</td>
</tr>
</tbody>
</table>

*Table 3.1: Numbers of edges in NAPAbench networks*

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. Elegans</em></td>
<td>2805</td>
<td>4495</td>
</tr>
<tr>
<td><em>D. Melanogaster</em></td>
<td>7518</td>
<td>25635</td>
</tr>
<tr>
<td><em>H. Sapiens</em></td>
<td>9633</td>
<td>34327</td>
</tr>
<tr>
<td><em>S. Cerevisiae</em></td>
<td>5499</td>
<td>31261</td>
</tr>
</tbody>
</table>

*Table 3.2: Number of nodes and edges in Isobase networks*
Our evaluation metrics differ for our two databases. For the Isobase dataset, which consists of experimentally-derived PPI data, we evaluate alignment quality using ICS, LCSC, and GOC. For the NAPAbench synthetic dataset, which was produced using models of PPI network evolution, we know what the true alignments of the given networks are. This makes evaluation much easier. Instead of reporting GOC, we report the percentage of aligned pairs that are correct. We also report ICS and LCSC for the NAPAbench data, as well, so that we can compare and contrast aligner performance on these noise-free synthetic datasets versus the real-world PPI data from Isobase.

3.1.2 Alignment Algorithms Evaluated

Our primary focus in this chapter is to evaluate and compare the many alignment programs that have cropped up in the past few years. There are many algorithms that have been published since 2010 that have never been directly compared to one another. Many of these recently published network alignment papers use only the older IsoRank [88] or GRAAL [57] for a baseline, or are additionally compared to MI-GRAAL [58]. Since IsoRank and GRAAL were among the first biological network aligners created, it is unsurprising that more recent algorithms perform better. Furthermore, differences in evaluation datasets used makes it difficult to even compare algorithms transitively based on their performance relative to these older algorithms. The goal of this Chapter is to provide direct comparisons of these algorithms so as to better inform practitioners who might want to use such alignment software in their own work, as well as to illuminate differences in relative aligner performance so that we can better understand the performance we need to attain to be competitive when we present our new algorithms in Chapter 5.
A number of criteria determined the selection of algorithms to include. First, given how many new algorithms have been presented in the past few years, more recent algorithms are favored. Furthermore, we tend to favor algorithms that are presented as tools—those which are accompanied with publicly available software that can be used relatively easily, typically providing a command line interface. Given these restrictions, as well as time restrictions and difficulties with getting some programs to run at all, we have omitted several tools that could be competitive with the ones discussed here [8, 39, 45, 51, 54, 66, 76, 90, 92]. We also omit MAGNA [83] and PISwap [14] for now, though we will compare our new algorithms to these two in Chapter 5.

Most of these alignment algorithms include a large number of parameters the user must set when invoking them. Wherever possible, we used the parameter settings reported to be optimal by the authors of these algorithms, or the settings suggested in the documentation accompanying the aligners. The arguments provided to each aligner is provided in Table 3.3.

These aligners also varied greatly in their total running times. Since we executed these aligners in several different virtual machines, we do not report their exact running times, since they have likely been skewed by virtualization, but we do report roughly how long they each took in Table 3.4.
<table>
<thead>
<tr>
<th>Aligner</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETAL</td>
<td>-a 0.0001 -b 0 -c 0.5 -i 2</td>
</tr>
<tr>
<td>PINALOG</td>
<td>No user-provided parameters.</td>
</tr>
<tr>
<td>SPINAL mode 1 and 2</td>
<td>alpha = 0.7</td>
</tr>
<tr>
<td>GRAAL</td>
<td>alpha = 0.8</td>
</tr>
<tr>
<td>C-GRAAL</td>
<td>No user-provided parameters.</td>
</tr>
<tr>
<td>MI-GRAAL</td>
<td>-p 23</td>
</tr>
<tr>
<td>GHOST</td>
<td>For the extractor, k = 3, except for <em>H. Sapiens</em>, where the extractor ran out of memory, in which case we used k = 2. For the aligner, matcher: linear, nneighbors: all, beta: 1.0, ratio:8.0, searchiter: 10.</td>
</tr>
<tr>
<td>NATALIE</td>
<td>-if1 5 -if2 5 -of 9</td>
</tr>
<tr>
<td>IsoRank</td>
<td>-K 30 -thresh 1e-4 -alpha 0.9 -maxveclen 1000000</td>
</tr>
</tbody>
</table>

Table 3.3: Parameter settings used in aligner evaluation
<table>
<thead>
<tr>
<th>Aligner</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETAL</td>
<td>seconds</td>
</tr>
<tr>
<td>PINALOG</td>
<td>minutes</td>
</tr>
<tr>
<td>SPINAL</td>
<td>minutes</td>
</tr>
<tr>
<td>GRAAL</td>
<td>hours to days</td>
</tr>
<tr>
<td>C-GRAAL</td>
<td>hours to days</td>
</tr>
<tr>
<td>MI-GRAAL</td>
<td>hours to days</td>
</tr>
<tr>
<td>GHOST</td>
<td>hours to days</td>
</tr>
<tr>
<td>NATALIE</td>
<td>minutes</td>
</tr>
<tr>
<td>IsoRank</td>
<td>minutes</td>
</tr>
</tbody>
</table>

Table 3.4: Typical running times of evaluated aligners

3.2 Results and Discussion

3.2.1 Synthetic Test Results

Benchmarking with the synthetic NAPAbench data produced very interesting results, showing a wide diversity in algorithm and software quality. First of all, we must note that GRAAL crashed on one of the CG problems, but ran successfully for the other 8 tests. When we report averages for this aligner, we average only over the alignments for which it ran to completion.

Induced conserved structure (Figure 3.1) is the first metric we consider. NATALIE attains the best score, significantly ahead of GHOST, SPINAL, and MI-GRAAL, which
Figure 3.1: ICS for all aligners with the NAPAbench benchmark data. For each aligner, we report the average for each of the network evolution models used. We also report the average over all the models. Note: in all of these bar charts, the order of aligners in the legend is the same as the order of the bars in the chart.

are otherwise the aligners with the highest ICS scores. This can partly be explained by NATALIE’s alignment strategy, which leaves nodes unaligned when it cannot find a good match. This in turn reduces the size of the denominator in the ICS measure. Aside from NATALIE’s high performance on this measure, the results for the other aligners show that C-GRAAL, NETAL, PINALOG, and IsoRank have very similar performance, but they are still weaker overall.

Next, in Figure 3.2, we examine how many nodes of the smaller graph are included in the largest shared component found between the two graphs. This metric correlates
Figure 3.2: Number of nodes in the largest common shared component of the aligned networks, reported as the share of the nodes in the smaller network, for the NAPAbench dataset.

strongly with ICS on this dataset. MI-GRAAL, SPINAL, and GHOST find the largest shared subgraphs overall. C-GRAAL, NATALIE, PINALOG, and NETAL, while finding similarly-sized subgraphs when compared to each other, lag behind SPINAL and GHOST by a fair margin. Once again, GRAAL trails behind significantly. Notably, NATALIE is not among the top aligners for this metric. This is a result of NATALIE’s alignment strategy; aligning fewer nodes overall diminishes the size of the largest shared component.

To assess the biological relevance of the alignments produced, we examine how well the aligners align nodes to orthologs (Figure 3.3). Since we are using synthetic data from protein network evolution models, we know which proteins are truly orthologous and can easily calculate the percentage of aligned nodes which have been matched with their orthologs.
Here, we see that NATALIE manages astounding performance, with 99% of aligned nodes mapped to true orthologs on average. However, we must reiterate that NATALIE produces partial alignments; on these alignment problems, NATALIE aligned on average 86% of the nodes in the smaller graph and left the rest unaligned. Even taking this account, the total number of correctly-aligned nodes is highest for NATALIE. SPINAL, PINALOG, and GHOST are essentially tied for second place with between 75% and 80% of nodes aligned to true orthologs. IsoRank and MI-GRAAL follow slightly behind these, with C-GRAAL yet further behind, and GRAAL performing quite poorly. This is an extremely important metric, since it represents the ultimate goal of biological network alignment—identifying orthologous proteins.
3.2.2 Real-World Test Results

We find that the relative performance of these algorithms is strikingly different when tested on real PPI data from the IsoBase data set [72]. In these tests, we attempt to align PPI networks from *C. elegans*, *D. melanogaster*, *S. cerevisiae*, and *H. sapiens*. For these tests, we get a much wider range of results, and poorer results overall. We suspect that much of this variance is due to the amount of noise present in experimentally-derived data.

As before, we experienced difficulties with certain programs crashing. GHOST crashed on two problems. SPINAL mode 1 crashed on two inputs, and GRAAL and MI-GRAAL crashed on three. C-GRAAL appeared to run to completion for all problems, but produced no output for two. Thus, missing bars in these figures implies that the corresponding program failed to run. As before, when averages are reported, they are averaged over only those alignments which the given aligner completed successfully.

On the *ICS* metric, we see that NETAL outperforms the other algorithms by a fair margin, while NATALIE, MI-GRAAL, and GRAAL perform similarly (Figure 3.4). Notably, NATALIE performs much better with *ICS* than *EC* on these tests. While NATALIE aligns fewer nodes, those it does align are aligned quite accurately, into parts of the second graph that they fit well. Oddly, given its performance elsewhere, we see GRAAL performing rather well, at times outperforming C-GRAAL, and in one case outperforming GHOST. This can be attributed to the fact that, like NETAL, GRAAL is not attempting to maximize a tradeoff of both topological and sequence similarity, but instead attempts to maximize topological fit only.

For the largest connected shared component, NETAL once again leads in the percentage of both nodes and edges of the smaller graph that are found as a connected component in the larger (Figure 3.5). In its best alignment, it finds that 90% of the nodes and 32%
of the edges in the *D. melanogaster* network are contained in a subgraph of the *H. sapiens* network. MI-GRAAL also produces excellent LCSC scores, and SPINAL and C-GRAAL also perform well. PINALOG, SPINAL in mode 1, GRAAL, NATALIE, and IsoRank all produce alignments with LCSC sizes so small as to be negligible. It is notable that NETAL and GRAAL perform so differently here, since they both attempt to maximize topological similarity alone. Between this result and the ICS results, it seems that NETAL is the superior solution for constructing topology-only alignments.

Next, we consider GOC scores. First, we consider GOC when we include all GO terms at depth 5 from the root of their ontologies (Figure 3.6). This gives us a greater number of GO annotations to work with, but, since most aligners use sequence similarity
Figure 3.5: Share of nodes in the smaller network in the largest connected shared component found for each alignment.

to construct the alignment, this inflates scores by including GO terms that were assigned based on sequence similarity themselves. Instead of being the top performer as it was on the topological benchmarks, NETAL’s performance here is the poorest. PINALOG and SPINAL mode 1 dominate the chart and IsoRank performs ably, while other alignment algorithms vary widely in performance. For the alignments it successfully produced, GHOST achieves $GOC$ scores significantly lower than PINALOG and SPINAL mode 1, but significantly higher than SPINAL mode 2, GRAAL, MI-GRAAL, C-GRAAL, and NATALIE, which all produce relatively poor $GOC$ scores on most of these tests. The generally lower performance on the $C.\ elegans$ problems appears to be due to the small size and extreme
Figure 3.6: GO Consistency scores for each alignment of the IsoBase PPI networks.

sparsity of the *C. elegans* network in the IsoBase dataset. This network has only half the nodes of the next-largest network, and only 14.4 % of the edges.

We also computed *GOC* scores using only GO terms with experimental evidence codes. Since, admittedly, GO terms are already transferred successfully on the basis of sequence similarity alone, excluding such terms from our set of evaluation GO terms gives us different information. If GO terms have been attributed to a protein simply due to its sequence similarity to another protein, then we would expect to see such GO terms match when we perform network alignment that uses sequence similarity information. While it is useful to see in Figure 3.6 that that indeed occurred, it is also helpful to see that experimentally-verified GO annotations are also shared between aligned proteins. We found that with this restricted set of GO terms, all *GOC* scores are much lower, reflecting the
lower number of experimentally-verified terms compared to the total number of terms in the GO database, but the relative ranking of the algorithms remains the same. These results are very similar to those in the paper where GOC is introduced [3]. Since the relative performance of aligners is not different when counting GOC in this way, we do not include bar charts for these results.

3.2.3 Comparing Overall Performance

Looking at results from both synthetic and real-world benchmarks, we are left with a dizzying variety of metrics. As particularly observed on the real-world data, some algorithms perform extremely well with respect to one metric, while performing abysmally with respect to another. This is a consequence of the two metrics by which these aligners perform their matching. Since they use both topological similarity information as well as biological similarity information to perform their alignments, each aligner must be designed to produce a good tradeoff between these two goals. Following Patro and Kingsford [74], we plot the topological and biological quality of these aligners against each other, so as to reveal the aligners that best jointly optimize topological and biological similarity in producing their alignment.

For the NAPAbench testing data, we plot $ICS$ against the percentage of nodes aligned to true orthologs (Figure 3.7). Doing so, we can quickly pick out the best performing algorithms by looking at which ones tend toward the upper-right of the graph. We see that NATALIE performs best, with GHOST and SPINAL yielding similar results to each other and lagging behind NATALIE. MI-GRAAL, PINALOG and C-GRAAL also perform reasonably well, but NETAL and GRAAL show poorer alignment results.
Figure 3.7: Plot of average percentage of nodes correctly aligned vs. integrated conserved structure score for all aligners with the NAPAbench benchmark data.

The results for the real-world PPI networks, on the other hand, are more ambiguous (Figure 3.8). First of all, we must emphasize that this scatter plot cannot be directly compared to the NAPAbench plot, because the y-axis here is our GO Consistency score, whereas with NAPAbench, we report the percentage of nodes aligned to true orthologs. However, we can still compare these two datasets with respect to the relative performance of different aligners, and examine the tradeoffs between the topological and biological quality of the resulting alignments. While SPINAL mode 1 manages the highest GOC score, and NETAL attains the highest ICS, those that manage a decent tradeoff between the two tend to perform poorly overall. If we must declare a best performer for these benchmarks, we might, hesitantly, point out GHOST, NATALIE, MI-GRAAL, and SPINAL mode 2.
Figure 3.8: Plot of the average GO consistency vs. average integrated conserved structure score for all aligners with the IsoBase network data.

as aligners that find alignments with decent topological and biological similarity results. However, compared to the synthetic data of the NAPAbench test set, all these algorithms experience much greater difficulty with these PPI networks derived from experimental data.

The difference between these aligners’ results on NAPAbench and IsoBase benchmarks is striking enough to deserve additional comment. All aligners perform much better on NAPAbench overall. While the most likely difference is the fact that NAPAbench networks contain no spurious or missing edges, we must also keep in mind the possibility that the PPI network evolution models used to produce NAPAbench’s networks may also contribute to the differences in performance. It’s possible that the networks generated by these models are easier for existing aligners to align, compared to real PPI networks. NAPAbench’s networks
Figure 3.9: Plot of the GO consistency vs. integrated conserved structure score for aligners with user-adjustable tradeoff between topological and biological similarity. We explored the full range of tradeoff levels for these aligners on the C. elegans - D. melanogaster alignment problem. This plot shows that the tradeoff parameters for these aligners cover different, non-overlapping regions of the objective space.

Several aligners support adjusting the relative importance of topological and sequence similarity when constructing their alignment. Of the aligners we benchmarked, SPINAL,
GHOST, IsoRank, and NATALIE support this functionality. We ran these aligners with differing settings to the tradeoff parameter repeatedly on the *C. elegans* - *D. melanogaster* alignment problem (Figure 3.9). These settings covered the full range of these aligners’ tradeoff parameters. The results are similar overall to Figure 3.8. We see a rough Pareto front consisting of results from SPINAL mode 1, GHOST, and NATALIE. SPINAL mode 2 results in lower performance than GHOST, and IsoRank is dominated by both GHOST and SPINAL mode 1. It is important to point out that even when we vary the tradeoff of these aligners through their whole range, the absolute differences in the alignments produced by a single aligner are small. We do not see any one aligner that covers a large portion of the frontier, but instead see that each aligner covers a small section of it. This indicates that, to get a good idea of the variety of alignments possible between two species, it is necessary to use multiple aligners and compare their results. These results inspire us to use a multiobjective algorithm in Chapter 5, so that we can explicitly explore the possible tradeoffs within this objective space.

### 3.3 Benchmarking Conclusions

In the past few years, many new algorithms for pairwise global network alignment have appeared, but few of them had been compared directly. By benchmarking these algorithms on synthetic and real-world data, we have shed light on the relative behavior and performance of these algorithms. The results have been surprising, and show that these algorithms can behave very differently on different data sets. Several algorithms perform very well, but the great differences in behavior from one test to another, and the tendency for some of the existing programs to crash, makes it difficult to recommend any one aligner. Those who want to use network alignment as part of their work should try a few of the
better-performing algorithms found here, and see how their performance compares for the particular data set in question. We suggest SPINAL, NATALIE, and PINALOG as good aligners to try at first; they produce their alignments very quickly while giving competitive results. GHOST is also an excellent performer, but its high memory requirements, slow speed, and crashes render it more difficult to use. However, GHOST is currently being rewritten in C++, and this may change in the near future. C-GRAAL and MI-GRAAL may be worthwhile in some situations, though other aligners perform better in general. All members of the GRAAL family share a bottleneck in the unpredictably slow graphlet-counting step, and this step also tends to crash. Since NETAL does not currently use sequence similarity in constructing alignments, its uses are more limited, but it can produce much higher ICS scores and larger connected components than other aligners on a given dataset, which may be useful in understanding the relative performance of other aligners. We cannot recommend GRAAL and IsoRank, as they are bested by many aligners on both GOC and ICS scores. This is understandable, given that most of these other aligners were benchmarked against GRAAL and IsoRank when they were being designed.

We have also found a number of issues that show the need for the work we introduce in later chapters. Sensitivity to noise is a major problem with existing aligners. Given the drastic differences in aligner performance between the noise-free synthetic data and the noisy real-world data, it is clear that future alignment algorithms must become even more robust to such difficulties. We have also shown that existing aligners produce alignments in non-overlapping portions of objective space, and each individual aligner covers only a small portion of the total region of objective space covered by all aligners together. Last of all, we have seen that many aligners perform well at either yielding good topological or good
biological matches, but few do both well. We make significant progress on these issues in Chapter 5.
CHAPTER 4

MACHINE LEARNING-BASED ESTIMATION OF NODE SIMILARITY

4.1 Introduction

The primary goal of this thesis is to leverage our evaluation of existing aligners in order to introduce a multiobjective genetic algorithm for PPI network alignment. We were initially concerned that, while multiobjective optimization has the potential to produce higher-quality alignments, it may be significantly slower than other solutions. A common way of solving this problem in genetic algorithm design is to seed the initial population with solutions that are closer to optimal than if they had been constructed at random [65]. This can cause the search to converge much more quickly. We investigate using machine learning to create these initial solutions. Specifically, the classifier will be asked to identify whether a given pair of proteins from two PPI networks should or should not be aligned to one another.

We found that this step, while producing better-than-random alignments, was actually slower than simply using multiobjective optimization directly, as will be detailed in Chapter 5. We also could only apply this approach to synthetic networks, since this was the only labeled data available. For these reasons, we ultimately abandoned using machine learning in creating alignments. While this approach was unsuccessful, we include this brief chapter as a preliminary report on what might still be a fruitful avenue of exploration in the future.
4.2 Data Used

Perhaps the simplest machine learning task is supervised learning, where the learning algorithm’s task is to learn from labeled examples. While this is generally the easiest to use and evaluate, it is dependent on the availability of labeled data. In the case of PPI network alignment, there is very little data we can use as labeled data– the research topic is so new that there are no canonical network alignments that are widely considered correct. Since we want to begin with supervised learning for simplicity, we therefore considered two alternatives for acquiring labeled data.

- Create many alignments of existing networks, and label them by the widely-accepted metrics for evaluating alignment quality. Instead of labeling each alignment as “right” or “wrong”, we assign numerical scores that just relate how good the alignment is, without knowing how high these numerical scores would be for a perfect alignment.

- Use fake data that resembles real PPI network data, but for which the true alignment is known.

Since synthetic PPI networks are available and have been proposed for benchmarking PPI network alignment algorithms, we chose the latter approach. We used the NAPAbench [81] set of PPI network alignment problems, as we did in Chapter 3 for benchmarking. The only downside to using this data is that it is perfect– real PPI networks are still incomplete and often have missing nodes and edges, as well as spurious edges. Therefore, while the classifiers we create are interesting preliminary work, we do not expect them to generalize to real PPI networks.

We created our data set by running a large number of existing alignment algorithms [3,27,57,58,64,77,88] on these test problems, creating dozens of alignments in total. Then,
we computed a number of graph-theoretic topological metrics for each node of the synthetic networks. We then created a large file where each observation is an aligned pair, and the features are the differences between each of these topological metrics for the given pair. This yielded a total data set of 795,649 rows and 19 columns, spanning dozens of alignments for 30 separate synthetic alignment problems. Our target variable is whether or not the given aligned pair is correctly aligned. We used Weka [34] to explore the data and evaluate classifiers.

We used the following features for each aligned pair of nodes. To keep this chapter brief, we generally give a non-technical description of each graph-theoretic metric. Our primary goal here is to evaluate how these “off-the-shelf” metrics available in any graph library can be used to aid in network alignment. The graph analysis library NetworkX [33] was used to produce this data, and its documentation was also consulted for the following descriptions of the features.

1. **Closeness centrality:** Intuitively, this measures the amount of time it will take to propagate information from a given node to the rest of the graph. The closer a node is to all other nodes, the higher its closeness centrality.

2. **Betweenness centrality:** This measures how many shortest paths a given node lies on. In other words, if a given node lies along the shortest routes between other nodes in the graph, it has a high betweenness centrality, since it lies between more nodes.

3. **Eigenvector centrality:** Another measure of how central to the graph a given node is. This one is defined in a recursive manner: a node is central if its neighbors are also central.
4. **Communicability centrality:** The sum of the lengths of all walks starting and ending at the given node. The intuition here is that if this number is higher, it's easier to move around the graph starting from the given node.

5. **Load centrality:** Similar to betweenness centrality, this is the proportion of all shortest paths in the network that pass through the given node.

6. **Estrada index:** This is the sum of the eigenvalues of a graph. We assigned to each node the Estrada index of its 3-hop neighborhood. This index was originally designed with the intent of reducing an entire graph to a single number such that similar graphs were assigned similar numbers. This is used as a feature in cheminformatics, where graphs represent molecules.

7. **Assortativity:** This measures to what extent the degree of a node determines which other nodes it is adjacent to in the graph. We computed this for each 3-hop neighborhood of each node to transform this into a per-node metric.

8. **Average neighbor degree:** Simply the average number of neighbors that the neighbors of the given node have. This measures the density of the network around a given node.

9. **Triangle count:** Count of the number of triangles that occur in the 3-hop neighborhood of the given node. The hope here is simply that this will help in giving each node a more unique fingerprint for the classifier to identify.

10. **Transitivity:** For the 3-hop neighborhood of the given node, counts how many triangles exist and divides it by the maximum number of triangles that could possibly exist among that many nodes.
11. **Clustering:** For a given node, the fraction of possible triangles that exist involving that node. This is very similar to transitivity.

12. **Square clustering:** Like clustering, but using squares instead of triangles.

13. **Pagerank:** The PageRank metric for the given node. This metric was originally used to rank the importance of web pages by their place in the topology of the web [71].

14. **Hubs metric:** Another search engine metric, from the HITS algorithm [47]. Estimates the value of the node based on the number of outgoing links from it.

15. **Authorities metric:** Also from the HITS algorithm. Estimates the value of a node based on the number of ingoing links to it.

16. **Degree:** The number of edges incident to the given node.

17. **Density:** The fraction of possible edges that exist in the 3-hop neighborhood of the given node.

18. **BLAST bit score:** The sequence similarity between the RNA sequences of the two proteins, as measured by BLAST [4].

We then use the absolute value of the difference of these metrics between each aligned pair in our data set as features, except for BLAST bit score, which we use directly.

### 4.3 Selecting Features

Our initial strategy was to include all the features we could easily extract given the network analysis library we chose to use, even though it seemed likely that many of these features would be highly correlated. Visual inspection of scatter plots confirmed this suspicion: the hubs and authorities metrics were perfectly correlated since our graphs were
Figure 4.1: Weka plot of eigenvector centrality (x-axis) vs. authorities metric (y-axis) for a subsample of the dataset. The red points are correctly-aligned pairs, which appear to cluster toward 0 for these metrics. This was one of the more interesting 2D scatter plots contained in the data.

undirected, and eigenvector centrality correlated with those two metrics as well. Interestingly, most of the centrality metrics did not correlate well at all, and so seemed to be getting at different information.

We include two of the more interesting scatter plots in Figs. 4.1 and 4.2 here. These show that there is at least some pattern in the features we have selected that may be useful for classification. None of the possible scatter plots showed that the classification problem would be easy: there were no linearly-separable clusters of correctly and incorrectly aligned pairs, or other neat ways to divide the data as we want to. This suggested that we would
be best off using classifiers that are capable of finding complex non-linear patterns in the data.

For feature selection, we first used the $\chi^2$ test for dependence between each feature and the target feature. This metric is fast to compute and gives us a good ranking of the features. Many classifiers increase greatly in computational complexity as the number of features increases, so we wanted to decrease the number of features early in our work. The results of this feature selection run are shown in Fig. 4.3.

Based on the results from the $\chi^2$ attribute selector, we removed all of the attributes that were given a score of 0, except for bit score—since this is the only biological attribute in our dataset, we wanted to investigate it further. The drawback of the $\chi^2$ selector is that it considers each attribute individually, without checking for strong correlations with other

Figure 4.2: Weka plot of PageRank (x-axis) vs. assortativity (y-axis) for a subsample of the dataset. The red points are correctly-aligned pairs, which form interesting clusters in this plot.
Ranked attributes:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>795</td>
<td>assortativityDiff</td>
</tr>
<tr>
<td>179.4994</td>
<td>closenessCentDiff</td>
</tr>
<tr>
<td>95.288</td>
<td>eigenvectorCentDiff</td>
</tr>
<tr>
<td>80.3081</td>
<td>hubsDiff</td>
</tr>
<tr>
<td>80.3081</td>
<td>authsDiff</td>
</tr>
<tr>
<td>72.9715</td>
<td>localDensityDiff</td>
</tr>
<tr>
<td>66.8593</td>
<td>transitivityDiff</td>
</tr>
<tr>
<td>52.6601</td>
<td>avgNbrDegDiff</td>
</tr>
<tr>
<td>21.2183</td>
<td>sqClusteringDiff</td>
</tr>
<tr>
<td>21.0056</td>
<td>loadCentDiff</td>
</tr>
<tr>
<td>20.6413</td>
<td>betweennessCentDiff</td>
</tr>
<tr>
<td>20.4508</td>
<td>pagerankDiff</td>
</tr>
<tr>
<td>0</td>
<td>commCentDiff</td>
</tr>
<tr>
<td>0</td>
<td>degreeDiff</td>
</tr>
<tr>
<td>0</td>
<td>bitScore</td>
</tr>
<tr>
<td>0</td>
<td>estradaDiff</td>
</tr>
<tr>
<td>0</td>
<td>clusteringDiff</td>
</tr>
<tr>
<td>0</td>
<td>trianglesDiff</td>
</tr>
</tbody>
</table>

Figure 4.3: Results of Weka’s chi-squared attribute ranker. Assortativity is the most useful attribute by far, followed by closeness and eigenvector centrality.

attributes. For this reason, we also removed the authorities metric, since it was perfectly correlated with the hubs metric and thus added no new information.

After removing a good number of features in this way, we then proceeded to a more computationally expensive form of feature selection: principal components analysis [41]. This resulted in a top-ranked vector consisting of load centrality, betweenness centrality, PageRank, the hubs metric, and eigenvector centrality. We decided to use this set of attributes when performing clustering and classification.

4.3.1 The Question of Bit Score

All classifiers perform extremely well when bit score, a measure of biological similarity, is included. We are hesitant to include this attribute in our classifiers, however. The bit scores given are, like the rest of our data, synthetic. Given how differently aligners performed on synthetic vs. real data in Chapter 3, we have no reason to believe that they will form
a similar distribution in real biological data. Thus, we focus on graph-theoretic attributes.

After all, graph isomorphism is graph isomorphism regardless of whether the networks were formed by nature or by a computer program, so we expect graph theoretic metrics to generalize much better. Therefore, while we could get very high accuracy on this synthetic data by using bit score information, we omit it, expecting the accuracy we get without it to be more realistic. Good alignments need to use both topological and biological similarity information, so an eventual classifier for real networks should use bit score data as well.

4.4 Supervised Learning

Nearly all existing biological network alignment algorithms require a function that takes a pair of nodes as input and outputs the confidence that aligning those two nodes would contribute to a good alignment overall. To the best of our knowledge, no work has been done on learning such an estimation function from common graph theoretic metrics. Our goal is to experiment with different learning algorithms to see if they can be used for this purpose. In all of the experiments below, we used Weka algorithms with 10-fold cross validation on a set of 795 data points.

4.4.1 Without Bit Scores

In classification, it is often wise to begin with the simplest classification algorithms available. Such algorithms are easy to understand, fast to compute, and, often, less likely to overfit or “memorize” the data they are trained on. Therefore, we began with rule-based learning algorithms. These algorithms try to find a simple if-then rule that correctly classifies the data. For instance, the OneR classifier [38] tries to find a rule that uses one attribute. OneR in this case was 65% accurate using a rather large conjunctive rule that
split based on eigenvector centrality difference. Another conjunctive rule classifier, aptly named ConjunctiveRule, managed 62% accuracy with a much smaller rule: if the eigenvector centrality difference is greater than 0.002305, then the pair should not be aligned. The more complex JRip rule learner [17] produced a set of rules similar in complexity to a decision tree and managed 68.8% accuracy. These performance levels are rather modest, and so we moved on to other methods in the search for better results.

Perhaps simpler than producing a propositional rule to classify data is to store the data and then, when confronted with a new data point, label it with the label of the closest stored data point. This is what the IB1 nearest neighbor classifier does [2]. This classifier is extraordinarily fast at training (which is just storing the instances in a data structure) and classifying (querying the data structure). This classifier managed a promising 85% accuracy on this data set. This looks to be a particularly promising technique for alignment, since it is fast, easy to implement, and manages reasonable performance.

Next, we tried decision trees. We expected their performance to be similar to but somewhat better than the rule-based classifiers, since decision trees are essentially hierarchical rule-based classifiers themselves. J48 [79] managed accuracy of around 68% regardless of the settings used. RandomTree, which considers attributes in a random order as it is built [12], managed to do a little better with an accuracy of 70.4% when limited to a maximum tree depth of four. Other trees performed similarly, with accuracy of around 70%.

We also considered multilayer perceptrons [80] and support vector machines [36]. These classifiers are often applied to very difficult learning tasks. However, on this data set, they performed worse than simple rule-based classifiers, with a multilayer perceptron being 54% accurate and the best SVM being 52% accurate.
Lastly, we considered several metaclassifiers. Such classifiers generally work by using a collection of very simple classifiers, and somehow combining their results. For instance, AdaBoost [32] works by performing a weighted sum of many small decision trees, while also weighting training example by their difficulty, to focus learning on difficult examples. This can often increase performance significantly. However, on this dataset, AdaBoost only managed a disappointing 70% accuracy. However, a random forest classifier [11], which constructs a set of random trees and then takes the class these trees vote for as its answer, managed an impressive 84% accuracy using only 10 random trees.

4.5 Building a Simple Aligner

The next task was to construct a simple aligner based on one of these classifiers. There were several important considerata for this task.

4.5.1 The Danger of Low Accuracy

There is a particular problem with using classifiers to select pairs of nodes to align. Assume we have a node \( v \in V_1 \) that we are trying to find a match for in \( V_2 \). Furthermore, assume \( |V_1| = 3000 \) and \( |V_2| = 4000 \). Then, if we have not matched any other nodes in \( V_2 \) yet, we need to execute our classifier for 4000 possible matchings – one for each node in \( V_2 \). We have been assuming there is one uniquely best node in \( V_2 \) for \( v \) to be matched to. If our classifier is 85% accurate, it will misclassify about 15% of \( V_2 \) as a good match. In other words, it will recommend 600 pairings, and we can’t even be sure that the correct pairing is among them! Worse still, when using our full data set (about 1000 times the size of the data set we gave to Weka) to our training sets and evaluating with Scikit-Learn [75], another machine learning library that can handle larger datasets than Weka, the best-performing
classifiers described above degrade in accuracy significantly, to around 62%. How much worse might they be when faced with real-world networks for which labeled data does not exist?

It may be that machine learning is not feasible for this problem given these attributes, but we must stress that these results discard bit score, a measure of biological similarity. When bit score is included, classifiers achieve performance of around 96%. However, when bit score is included, classifiers tend to use bit score alone in their classification, which, as we show in Chapter 5, would not work on real data.

4.5.2 Matching Algorithm

Once our classifier has given us its output of potential matchings, we still need to find a way to match one network to another. As mentioned above, the classifier may make many conflicting recommendations, so the matching stage of an aligner chooses which recommendations to take and which to ignore to construct the final alignment. Several possibilities exist.

The first possibility is to use the Hungarian Algorithm [68]. This algorithm constructs the maximum-weighted matching between two sets of nodes in $O(n^3)$ time. However, this algorithm was designed specifically for situations where the actual goodness-of-fit between each pair is known. In our case, we only have an imperfect estimate of how well a given pair of nodes can be matched. The prior literature has found that in this case, the Hungarian Algorithm can actually perform worse than asymptotically faster algorithms (see Chapter 3). Another possibility is to use very simple, fast heuristics like the 1/2 matching algorithm, which repeatedly aligns the highest-weight pair of nodes remaining [51]. One of the more common methods, though, is a so-called “seed-and-extend” algorithm which matches the
two most similar nodes, then matches their neighbors, and works outward until all nodes of the smaller network have been aligned. We decided to use the 1/2 matching algorithm and a seed-and-extend algorithm similar to that used by the existing aligner C-GRAAL [64].

4.5.3 Results

Since we had already implemented an alignment framework in Python, we relied on Scikit-Learn to implement the versions of our classifiers that we would use for the alignment process. We ultimately decided to experiment with two aligners: a nearest neighbor classifier that was trained on the five best topological features only, and an SVM that additionally used bit scores. These classifiers attained accuracies of 62.8% and 94% respectively. As noted before, bit scores considerably boost performance on the synthetic data set, but the classifiers that use them are less likely to generalize.

We first tried to use these classifiers in conjunction with the 1/2 matching algorithm. For each pair of nodes in $V_1 \times V_2$, we ran the given classifier to estimate the confidence that the given pair could be aligned successfully. We then used these confidence scores with the 1/2 matching algorithm to construct an alignment. Unfortunately, the results were abysmal. The ICS metric was essentially zero for both classifiers, with values well less than 0.001. For comparison, the best aligner on this dataset in our literature review, NATALIE 2.0 [27], achieved an ICS score of better than 0.98 on the same problem. The choice of classifier used with 1/2 matching did not substantially affect the results. Fewer than 0.1% of nodes were aligned to a correct ortholog.

Next, we tried a seed-and-extend algorithm that first aligns the highest confidence pair of nodes, and then aligns their neighbors, their neighbor’s neighbors, and so forth. Seed-and-extend algorithms are commonplace in the PPI network alignment literature because they
are faster than an optimal matching and tend to deliver good results. Because Scikit-Learn’s implementation of SVMs computed much faster than their nearest neighbors classifier, we only experimented with the SVM classifier with the seed-and-extend algorithm. Also, to give us a sort of baseline, we also used the seed-and-extend matcher in conjunction with randomly-generated similarity scores, since a seed-and-extend alignment is likely to produce many conserved edges anyway. The results were quite interesting: with random confidence scores, we obtained an ICS of 0.6145, but virtually none of the nodes were aligned to their true orthologs. With the SVM confidence scores, the seed-and-extend matcher achieved a slightly higher ICS of 0.6505, but it, too, failed to align nodes to true orthologs. This high ICS performance is consistent with how seed-and-extend works: even with random similarity scores, it would still bias towards matching neighbors to neighbors. In fact, when we were testing our seed-and-extend algorithm, seed-and-extend using only bit scores as a similarity measure outperformed IsoRank in both ICS and GOC.

Lastly, we tried to use the SVM classifier in conjunction with the Hungarian algorithm. In this case, the alignment had a somewhat modest ICS of 0.3302, but about 65% of the nodes in the smaller graph had been aligned to their true orthologs. This is perhaps the most promising result we had, since it successfully aligned the majority of nodes to a correct corresponding node in the other graph. The downside is that this matching process is prohibitively slow for our purposes, since we simply want a fast, approximate matching that is better than an entirely random one. Still, this shows that a machine learning classifier shows promise for use in network alignment, despite these results being poor compared to existing aligners.
4.6 Machine Learning Conclusions

In this chapter, we investigated whether various graph metrics for centrality, connectedness, and clustering could be used as input to a classifier that could identify whether given pairs of nodes should be aligned to produce a network alignment. We found that when such a classifier is combined with a computationally expensive matching method, non-trivial alignments can be found. Unfortunately, though, these results are still poor compared to the best existing aligners, and much slower, taking around 12 hours for the SVM similarity estimation combined with the Hungarian matching algorithm. Since some aligners can do better in minutes (see Chapter ??), this time expense is unacceptable.

However, further investigation may lead to better performance. Machine learning is, at heart, the estimation of functions, and we know from prior research into graph alignment that there are functions that can take two nodes and tell us whether they should be aligned. The features we used have not traditionally been used for these functions, but given the ease with which they can be computed and their presence in standard graph libraries, we thought it was worth trying them to see if it was possible to make a low-cost alignment algorithm.

A more fruitful future approach might be to use machine learning to analyze the previously successful measures of node similarity that have been used in graph alignment. Machine learning could be used to learn which of these estimation functions work best, or perhaps even learn that some combination of these estimation functions works better than any of them in isolation. This would be a more time-consuming task than the one presented here, because source code for these estimation functions is generally not available, so they would need to be reimplemented from the published pseudocode. Such an
investigation is certainly possible, however, and may yield interesting results. We discuss potential extensions to this preliminary investigation in Chapter 6.
CHAPTER 5
MULTIOBJECTIVE OPTIMIZATION FOR CREATING ALIGNMENTS

5.1 Introduction

In this chapter we present the primary contribution of this thesis— a new alignment algorithm that outperforms existing algorithms while producing hundreds of alignments at the same time. These alignments are spread across the tradeoff between biological and topological similarity, and better characterize this tradeoff than ever before. Since alignment algorithms are typically given names, we name our aligner \textit{Optnetalign}, an acronym for \textit{Optimizing Network Aligner}.

5.2 Multiobjective Optimization

Metaheuristics are a family of optimization techniques for problems that cannot be framed in a way amenable to standard optimization techniques, such as when objective functions are discontinuous, not differentiable, or not functions of real numbers. In PPI network alignment, our goal is to maximize several objectives that are a function of a network alignment – the problem must first be relaxed for standard optimization techniques to work [27]. Metaheuristics are thus a natural fit for network alignment. Genetic algorithms are a particular family of metaheuristics inspired by biological evolution [65]. Initially, a population of candidate solutions are generated at random. These solutions are ranked by their fitness, and the best of them are allowed to “breed” to produce a new population of solutions. The new population undergoes mutation with some small probability, and then
the process begins again, until an acceptable solution has been found [30]. This approach is applicable to many problems, since it only requires a way of scoring the “fitness” of candidate solutions, and a data structure representing solutions for which we can define a useful crossover operation (for combining two existing solutions) and a mutation operation (for randomly modifying solutions). Memetic algorithms are a generalization of standard genetic algorithms that additionally incorporate a local search heuristic to further improve their results [10]. Genetic algorithms and local search heuristics are complementary – the former explores large spaces quickly through crossover and mutation, while the latter is particularly effective at refining solutions that have been found.

While genetic algorithms have been used since the 1970s (e.g. [37]), variants capable of efficiently optimizing multiobjective problems have only become a focus of research more recently, with algorithms such as NSGA-II [24] and SPEA2 [96] becoming immensely popular. Other local search approaches to multiobjective optimization have had great success, as well [19, 22, 49]. Multiobjective memetic algorithms have also been developed in recent years [48, 50]. Multiobjective optimization is framed in terms of several concepts borrowed from welfare economics [23, 95]. The first is Pareto optimality: a solution to a multiobjective optimization problem is Pareto optimal if further increasing one of the objectives would decrease one or more of the other objectives. Such solutions represent optimal tradeoffs between the given objectives. The second concept is Pareto dominance: solution $x$ dominates solution $y$ if $x$ is at least as good as $y$ with respect to all objectives, and strictly better than $y$ with respect to at least one objective. This definition leads to the possibility that a set of solutions may be non-dominated with respect to each other. Multiobjective genetic algorithms work by selecting non-dominated solutions for reproduction at every generation, in the hopes of approximating the set of Pareto optimal solutions. This approximate Pareto
optimal set will, in turn, allow us to characterize the Pareto front of the objective space—the boundary between attainable and unattainable solutions. Multiobjective genetic algorithms output a number of representative solutions from the Pareto optimal set. Since these solutions are all Pareto optimal, we cannot a priori prefer one of them over another, and a human decision maker must use some other information or subjective preference to decide which of these solutions to keep. In the case of PPI network alignment, this means that our algorithm produces a number of alignments with different tradeoffs between topological and biological fit. These alignments can then be studied further to better understand the relationship between the two networks under consideration.

A problem shared by all existing aligners is that they do not explicitly approach the problem of network alignment as a multiobjective one. It has become increasingly clear that there is a tradeoff between our alignment objectives, as we discussed in Chapter 3. However, the best any existing aligner does to address this problem is providing users with a parameter $\alpha$, which gives the user rough control over the weight of biological or topological similarity in constructing an alignment, with some equation similar to

$$sim(u, v) = \alpha \times t(u, v) + (1 - \alpha) \times b(u, v)$$

(5.1)

where $sim(u, v)$ is the aligner’s estimated overall similarity between nodes $u$ and $v$ and $t(u, v)$ and $b(u, v)$ are the aligner’s estimated topological and sequence similarity of the two nodes respectively. This is insufficient. Not only does this approach requires trial-and-error experiments on the part of the user, and complete recalculation of the alignment each time $\alpha$ is adjusted, but existing aligners only cover a narrow section of the objective space, even when varying $\alpha$ as widely as possible. Optnetalign produces a variety of solutions more
diverse than existing aligners combined, and does so in a single run without extra effort on the part of the user.

5.3 Our Hybrid Algorithm

We present a multiobjective memetic algorithm that discovers a representative set of non-dominated alignments using crossover, mutation, and swap-based hillclimbing. Here, we detail the workings of our algorithm and our design rationale.

5.3.1 Alignment Representation

Following other genetic algorithms for network alignment, we adopt a permutation encoding for our alignments [7,83]. In this encoding, we label the $n$ nodes of each network with the integers $\{0, 1, \ldots, n - 2, n - 1\}$. Since we assume $|V_2| \geq |V_1|$, we add “dummy” nodes to $V_1$ so that $|V_1| = |V_2|$. Then our alignment $f$ can be stored in memory as an array $a$ such that if $f(u) = v$, then $a_u = v$. All indices $i \geq |V_1|$ of $a$ are ignored. This representation simplifies searching through the space of alignments to searching through the space of permutations of integers in the interval $[0, |V_2|)$, allowing us to adapt permutation-based search operators that have been well-studied in genetic and local search algorithms for the traveling salesman problem.

5.3.2 Crossover and Mutation

We adopt the Uniform Partially Matched Crossover (UPMX) operator that was used previously in a genetic algorithm for a related largest common subgraph problem [15]. We compared this to several other standard and custom-built crossover operators and found that it tended to give the best performance. UPMX works as follows: for two permutations
a and b, for each index i with probability \( cxswappb \), swap the value of \( a_i \) and \( b_i \). Unless \( a_i = b_i \), the new values of \( a_i \) and \( b_i \) after the swap will occur in their respective arrays in two places. To fix this, we also swap \( a_j \) and \( b_k \), where \( j \) and \( k \) are the indices of the duplicate values in their respective arrays. An example of this swap process for \( i = 0 \) follows.

1. Initial arrays: \( a = \{1, 3, 2\} \) and \( b = \{2, 1, 3\} \).

2. Swap the first elements of \( a \) and \( b \): \( a = \{2, 3, 2\} \) and \( b = \{1, 1, 3\} \).

3. \( a \) has a duplicate at \( j = 2 \) and \( b \) has a duplicate at \( k = 1 \). Swap \( a_j \) and \( b_k \): \( a = \{2, 3, 1\} \) and \( b = \{1, 2, 3\} \).

For mutation, we also adopt a simple swap-based scheme. For a permutation \( a \), for each index \( i \) with probability \( mutswappb \), we randomly select an index \( j \neq i \) and swap \( a_i \) with \( a_j \).

5.3.3 Local Search

We make use of two variants on a simple hill climbing algorithm based on swaps. The first variant, \( hillClimbNonDominated \), performs repeated random swaps, undoing a swap if it worsens any of the objective functions. The second variant, \( hillClimbOneObj \), takes an objective name as an argument, and only undoes a swap if the given objective is worsened, ignoring any worsening of other objectives. We use the former hill climb variant as a core function to improve our population of alignments, and use the latter to increase the variance of our population in objective space, which helps us to find a better set of representatives of the approximated Pareto front.

It is important to note that our crossover, mutation, and hill climbing functions all use swapping as their basic operation. Since the only way permutations are modified
is by swapping, this allows us to implement our fitness evaluations very efficiently—we only compute the change in the user-specified objectives at each swap. This allows us to evaluate millions of swaps per second on a desktop CPU. The $\text{hillClimbNonDominated}$ function alone can obtain results competitive with or better than the best previous aligners in seconds. The overall speed of the memetic algorithm is slower, since the population-based scheme causes some reduplication of effort among threads, but that is the price that must be paid to maintain a set of non-dominated solutions.

5.3.4 Archiving

Our population is stored in an archive data structure that automatically discards dominated solutions when better solutions are inserted. This gives us a data structure that stores only the non-dominated solutions that have been found so far. When the archive exceeds its user-defined size limit, it shrinks itself to the maximum size by discarding the most crowded solutions, using the crowded comparison operator first introduced in NSGA-II [24]. This helps to ensure that a diverse variety of solutions are maintained in the population at all times. This archiving strategy is similar to the multiobjective local search algorithm PAES [49].

5.3.5 Pseudocode

To increase the performance of our algorithm, it was designed from the beginning to be amenable to parallelization. The user specifies how many threads the program is to use, and each thread executes Algorithm 1 until a user-specified time limit has been reached. The threads all share the same archive, which is the only point of communication between the threads. This solution is highly scalable, and is able to fully utilize
16 processor cores in our tests. The overall structure of the algorithm is very straightforward, and reminiscent of other hybrid genetic algorithms in the literature; e.g. [70].

One quirk of the algorithm is that we first decide probabilistically whether to perform $\text{hillclimbOneObj}$ or $\text{hillClimbNonDominated}$, but then afterwards, perform another round of $\text{hillClimbNonDominated}$. This ensures that, when $\text{hillclimbOneObj}$ is chosen, the alignment is not left in a dominated state that could easily be improved to a non-dominated one.

The user-adjustable parameters of our algorithm include the time limit, the rate of crossover ($cxrate$) and mutation ($mutrate$), the probability of performing a swap at each index for crossover ($cxswappb$) and mutation ($mutswappb$), the probability that single-objective hill climbing will be used instead of non-dominated hill climbing ($oneobjrate$), and the number of iterations of hill climbing to perform in each loop ($nitors$). We find that over the course of the algorithm’s execution, the optimal rate of crossover, mutation, and one-objective hill climbing seem to vary. We thus provide the option of automatically adjusting those rates to their rates of success at producing non-dominated solutions over the course of the algorithm’s execution. Other, more sophisticated control schemes exist in the genetic algorithms literature [25, 26], but this simple heuristic appears to work quite well. We find that $cxswappb = 0.5$, $mutswappb = 0.0001$, and $hillclimbiters = 10000$ are good settings for the remaining parameters. Interestingly, we find that hill climbing is best used as the primary means for obtaining results, with crossover and mutation mostly playing a secondary role to prevent premature convergence.
Algorithm 1 Per-Thread Loop

\begin{algorithm}
\textbf{while} time limit not reached \textbf{do}

\((parent1, parent2) \leftarrow \text{two random members of Archive}\)

\textit{child} \leftarrow \text{initialize new alignment}

\textbf{with probability} \textit{cxrate} \textbf{do}

\textit{child} \leftarrow \text{crossover}(cxswappb, parent1, parent2)

\textbf{end}

\textbf{with probability} \textit{mutrate} \textbf{do}

\textit{child} \leftarrow \text{mutate}(mutswappb, \textit{child})

\textbf{end}

\textbf{with probability} \textit{oneobjrate} \textbf{do}

\textit{randObj} \leftarrow \text{randomly-chosen objective}

\textit{child} \leftarrow \text{hillclimbOneObj}(\textit{randObj}, \textit{niters}, \textit{child})

\textbf{else do}

\textit{child} \leftarrow \text{hillclimbNonDominated}(\textit{niters}, \textit{child})

\textbf{end}

\textit{child} \leftarrow \text{hillclimbNonDominated}(\textit{niters}, \textit{child})

\textit{insert}(\textit{child}, \textit{Archive})
\end{algorithm}

5.4 Results and Discussion

As discussed above, we report our results aligning the networks included in ISOBase [72], following the same methodology as in Chapter 3, except that we replace ICS with \(S^3\). For each alignment problem, we run our algorithm for 12 hours. We try two different sets of objectives. In the first experiment, we optimize \(S^3\) and the sum of the bitscores of aligned pairs. In the second experiment, we optimize \(S^3\) and \(GOC\). The first experiment is most
directly comparable to existing aligners, which try to optimize bitscore sum, while the second is an investigation of how well we can optimize the overlap of GO terms directly, which helps us obtain an approximate upper bound on the extent to which both topological fit and biological function can be jointly optimized. For each alignment problem, Optnetalign is instructed to output an approximate Pareto front containing at most 200 alignments.

Since Optnetalign outputs more candidate alignments than the other aligners we evaluate against combined, we report the average, minimum, and maximum of $S^3$ and $GOC$ of the output alignments for each problem. We also compare the front of alignments our algorithm produces to all the alignments produced by the aligners that have a user-configurable tradeoff parameter between topological and biological similarity for the *C. elegans* - *D. melanogaster* alignment problem.

5.4.1 First Experiment Results

Here we report the results of optimizing $S^3$ and the sum of bitscores of aligned pairs. Optnetalign matches or outperforms all existing aligners on $S^3$, and outperforms all but one aligner on $GOC$. As noted above, we report the minimum, average, and maximum of these scores. These metrics are almost perfectly inversely correlated, so for all of these experiments, the alignment Optnetalign produces with the highest $S^3$ has the lowest $GOC$, and vice versa.

We first consider our performance on $S^3$ in Figure 5.1. We must note that, as in our previous benchmarks in Chapter 3, several aligners crashed on several experiments. This is denoted by a missing bar in these instances. The maximum $S^3$ found by Optnetalign on each problem instance is comparable to or exceeds that of NETAL, which was previously the best performer on this data set. The only problem for which Optnetalign performs worse than
Figure 5.1: $S^3$ score for various aligners and our own for the Isobase dataset, when Optnetalign is set to optimize $S^3$ and sum of bit scores. Note that the bars in the chart are in the same order as in the legend.

NETAL is the *C. elegans* to *D. melanogaster* alignment problem, where NETAL attains an $S^3$ of 0.4117 and Optnetalign manages an $S^3$ of 0.396. On the *C. elegans* to *H. sapiens* alignment problem we attain the same $S^3$ as NETAL, with both aligners managing 0.3963. Despite this, NETAL and Optnetalign only agree on four aligned pairs out of 2,805! We will discuss the issue of alignment agreement further below. On the rest of these problems, our best topological alignments edge out NETAL. However, as shown in Figure 5.2, our topologically best alignments have much higher GOC scores than NETAL’s. Even our
alignments with the poorest topological quality tend to perform comparably to or better than many existing aligners.

With respect to GOC scores, our best alignments are all a distant second place behind PISwap. However, because PISwap works by first maximizing sequence similarity with the Hungarian algorithm, and then is limited to performing swaps that don’t worsen the sequence similarity matching, its topological fit is extremely low, with $S^3$ scores generally an order of magnitude lower than any other aligner. Thus, PISwap’s alignments constitute an extreme point on the Pareto front, though there may be some argument to be made as to whether an aligner that conserves so few edges can be considered to be performing network
alignment at all. For instance, on its topologically worst performance, PISwap only finds 91 conserved edges on the *C. elegans* to *S. cerevisiae* alignment problem, out of the 4,495 possible. Our best GOC scores, which outperform all aligners except PISwap, come from alignments that have non-trivial $S^3$ scores that are in some cases, such as the *S. cerevisiae* to *H. sapiens* problem, competitive with existing algorithms.

We also compare the Pareto front found by Optnetalign on the *C. elegans* to *D. melanogaster* problem to the range of tradeoffs produced by existing aligners that expose a tradeoff parameter to the user for balancing topological and biological similarity. The results are shown in Figure 5.3. These results are typical for all alignment problems we evaluated.
We show the scatter plot for this particular problem because the fewest number of alignment runs for the other aligners crashed on this problem. These results show that Optnetalign is able to find a much wider Pareto front than all previous aligners. Previously, it was not nearly as clear that such a wide range of results was possible. The alignments Optnetalign produced for this problem also varied greatly in the pairs of nodes they chose to align. The least similar pair of networks had only 8.7% of their aligned pairs in common. The maximum was 73%, and the average was 36%. This diversity in aligned pairs is comparable to the diversity in all the alignments produced by existing aligners.

5.4.2 Second Experiment Benchmark Results

Here we report the results we obtain when we set Optnetalign to optimize $S^3$ and $GOC$ directly. Since ours is the only aligner among those tested that can optimize GO term consistency instead of bit score sum, all other aligners are still optimizing bit score sum, when applicable, so all of the scores in Figures 5.4 and 5.5 are the same as in Figures 5.1 and 5.2, except for Optnetalign’s.

In Figure 5.4, we see that our $S^3$ results are much the same as before with the best alignments, with only slightly lower scores. The average scores are somewhat lower now, especially on the latter 3 alignments, where the average no longer outperforms most aligners. The minimum scores are very similar to those in Figure 5.1, as well.

The primary difference when optimizing $GOC$ directly, instead of its imperfect proxy of bit score sum, is that we are now able to obtain $GOC$ scores that outperform PISwap on several problems. While on the $C. elegans$ to $D. melanogaster$ problem, PISwap still strongly outperforms all other aligners, Optnetalign is actually able to outperform PISwap
Figure 5.4: $S^3$ score for various aligners and our own for the IsoBase dataset, when Optnetalign is set to optimize $S^3$ and GOC. Note that the bars in the chart are in the same order as in the legend.

on the latter three alignment problems, and performs much more strongly on the other problems involving *C. elegans*.

5.4.3 Comparing Objectives

The speed of Optnetalign, its ability to handle any number of objectives, and the large number of diverse alignments it can create in a reasonable amount of time, allow us to perform novel analyses that explore the inherent tradeoffs between many objectives, as well as examine how different objectives are correlated. This can give us insight into the nature of the network alignment problem, and help us identify deficiencies in current alignment
Figure 5.5: GOC for various aligners and our own for the IsoBase dataset, when Optnetalign is set to optimize $S^3$ and sum of GOC.

evaluation metrics. We examine this by performing another run of the $S. cerevisiae$ to $H. sapiens$ problem, in which we set the aligner to optimize $EC$, $ICS$, $S^3$, sum of bit scores, and $GOC$. This time, we set the maximum archive size very high, to allow the aligner to output as many non-dominated alignments as possible in a 12 hour run. This resulted in 571 unique, non-dominated alignments. The minimum agreement on aligned pairs for these alignments was 1%, the maximum was 92%, and the average was 8.3%. This lack of agreement is consistent with the lack of agreement between existing aligners [74], so this confirms that Optnetalign produces alignments as diverse as using many different previously-published aligners together.
<table>
<thead>
<tr>
<th></th>
<th>ICS</th>
<th>$S^3$</th>
<th>EC</th>
<th>GOC</th>
<th>Bit score sum</th>
<th>LCCS size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICS</td>
<td>1.0</td>
<td>-0.132</td>
<td>-0.338</td>
<td>0.079</td>
<td>-0.103</td>
<td>-0.42</td>
</tr>
<tr>
<td>$S^3$</td>
<td>1.0</td>
<td>0.957</td>
<td>-0.964</td>
<td>-0.848</td>
<td>0.939</td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>1.0</td>
<td>-0.948</td>
<td>-0.796</td>
<td>0.987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOC</td>
<td>1.0</td>
<td>0.81</td>
<td>-0.91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit score sum</td>
<td>1.0</td>
<td></td>
<td>-0.759</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCCS size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Correlation matrix for various objectives on the S. cerevisiae to H. sapiens alignment problem. Unlike the other objectives, largest common connected subgraph (LCCS) size was not set as an optimization objective in our program; it was computed afterwards.

We first present a correlation matrix for our objectives in Table 5.1. We can see that EC and $S^3$ are strongly positively correlated, but, as we noted above, since Optnetalign can minimize the ICS denominator without increasing the number of conserved edges, ICS is negatively correlated with the other two topological metrics. Interestingly, ICS correlates somewhat positively with GOC. Unfortunately, though, this is not because of some inherent advantage to ICS, but is instead an artifact of how Optnetalign cheats when optimizing ICS. For inflated levels of ICS, GOC begins to increase with ICS, but the alignments with the highest GOC actually have very low ICS scores, just as they have low EC and $S^3$ scores. The relationship between ICS and GOC illustrated in Figure 5.6. To better understand ICS inflation, we also present a plot of EC against ICS in Figure 5.7.

This experiment also demonstrates the potential utility of optimizing for GO term consistency instead of bit score sum. In Figure 5.8, we show that, while bit score sum and GOC generally correlate, the highest GOC scores we obtain have bit scores sums that are less than half of the highest bit score sum Optnetalign finds. This reinforces the fact that,
in some cases, proteins of similar function can have significant differences in their sequences, and it shows the value of optimizing GOC directly.

This large set of alignments again highlights the inherent conflict between optimizing measures of topological fit and measures of biological fit simultaneously. In Figure 5.9, we show $S^3$ plotted against GOC. Even though most of the alignments in Figure 5.9 dramatically outperform those of previous aligners, we still see a distinct tradeoff between $S^3$ and GOC. Since previous aligners varied dramatically in performance while only outputting
Figure 5.7: Scatter plot of EC vs. ICS on the S. cerevisiae to H. sapiens alignment problem.

a single alignment, there was very little evidence that such a wide Pareto front between these two objectives was obtainable. Previous aligners which contained a user-controlled tradeoff parameter between these topological and biological similarity scores framed the parameter as a small tweak, and in most cases, adjusting that parameter through its entire range, as we did in producing Figure 5.3, explores only a small portion of objective space. Therefore, it was not clearly understood that these two objectives conflict so sharply.
Figure 5.8: Scatter plot of GOC scores vs. sum of bit scores on the S. cerevisiae to H. sapiens alignment problem.
Figure 5.9: Scatter plot of $S^3$ scores vs. GOC scores on the S. cerevisiae to H. sapiens alignment problem.
5.5 Extracting Biological Data from Alignments: An Example and Difficulties

The high level of agreement in GO annotations in our resulting alignments gives us some confidence in the quality of the alignments produced. Previous work that attained lower GO agreement than Optnetalign reports many orthologous proteins being successfully aligned when performing alignments. While we mainly leave biological interpretation of these alignments to those qualified to make them, in this section we give one example of an apparently successful matching.

One of the organisms whose PPI network we use here, *C. elegans*, is widely used as a model organism for aging-related research, and one of the proteins of particular interest in the aging of this organism is AMP-activated protein kinase, abbreviated as AMPK in general. In *C. elegans*, the \( \alpha \) subunit of this protein, designated aak-2, has been found to be a key member of aging pathways in *C. elegans* [21]. Generally, AMPK is known for regulating cell metabolism based on available stored energy [35]. Since these proteins are well studied, we thought it would be interesting to verify that they are mapped correctly in our output. In our *C. elegans* to *H. sapiens* alignments, we investigate what human proteins aak-2 is mapped to. We find a high level of agreement among all output alignments for this particular protein. In virtually all of the first experiment results, where we optimize \( S^3 \) and bit score sum, the vast majority of alignments map aak-2 to PRKAA2, the corresponding \( \alpha \) subunit of human AMPK. When set to maximize \( S^3 \) and GOC, aak-2 is mapped primarily to MAPKAPK5, another closely-related protein.

Because we have so many alignments to consider, it is not easy to interpret how well proteins that interact with aak-2 have been matched to corresponding interacting proteins in *H. sapiens*. This is the main weakness of the network alignment research program exposed by our work– it is possible to create many alignments that do not completely agree with
one another, and how to interpret these alignments is an open question. In our dataset, aak-2 is involved in eight protein interactions, while PRKAA2 is involved in nine. There are \( \frac{9!}{(9-8)!} = 362,880 \) possible ways of matching eight proteins to nine proteins, and that is under the hopeful assumption that aak-2’s neighbors are only ever matched to PRKAA2’s neighbors! This puts into perspective the inherent difficulty of network alignment, as well as the inherent difficulty of visualizing and interpreting these alignments when many different and conflicting alignments are possible. The amount of data produced by one run of our algorithm is immense, and it is difficult to know where to begin in analyzing it. This is a problem that, so far, has not been adequately explored in network alignment, for two related reasons. First, it was not known that a tradeoff existed between topological and biological fit– it was thought that there existed one true alignment maximizing both. Second, because it was thought that there existed one best alignment, previous aligners only produced one alignment, which could be analyzed easily. We now know that there is an immense number of possible good alignments, and previous aligners only showed us one of them. Much more work must be done on interpreting these alignments now that this difficulty is known. However, the magnitude of this problem is such that it could easily be the topic of a thesis itself.

The disagreement between alignments optimizing bit score sum versus GOC also shows the need for further work that investigates which of these objectives is a better alignment objective. As we see above, GOC and bit score sum are not perfectly correlated, and this causes practical disagreements among alignments. We know that current GO annotations are not complete, and are incomplete in a way that is biased by what has been researched previously. Therefore, it may be best to optimize bit score sum, and use the results to help transfer GO annotations, instead of using aligners to maximize shared
GO annotations directly. We consider this to be another open question in PPI network alignment research that is in need of more exploration.

5.6 Multiobjective Optimization Conclusions

We have shown that Optnetalign can outperform or perform competitively with the best previously published aligners. Furthermore, since it is a multiobjective algorithm, it manages to better balance the tradeoff between the conflicting objectives of biological and topological fit. Its ability to produce a large number of alignments in one run allows us to explore the problem of network alignment more thoroughly, and reveals the strengths and weaknesses of current evaluation metrics. We expect Optnetalign will be useful not only in aligning particular PPI networks to one another, but also in evaluating and experimenting with new alignment objectives, as well. The large number of highly diverse alignments Optnetalign can produce allows us to thoroughly understand the sometimes complex relationship between alignment objectives, and also shows promise in developing new techniques to combine many alignments into summary alignments, to produce many-to-many alignments, or to create local alignments that find small-scale regions of similarity between PPI networks, as we will discuss in the next chapter.
CHAPTER 6

FUTURE WORK AND CONCLUSION

In this thesis, we have examined the problem of PPI network alignment, evaluated existing aligners, introduced our own aligner that achieves higher performance, and used our new aligner to illustrate some of the remaining difficulties in PPI network alignment. Here, we consider potential extensions of our work, as well as good next steps for attacking the problem of PPI network alignment in general.

6.1 Better Evaluation Metrics

Our results in the previous chapters show weaknesses in existing network alignment research that open up many possibilities for future work. Perhaps the most glaring question, given these results, is that of the tradeoff between biological and topological similarity. The assumption underlying the entire premise of global network alignment is that different species share large regions of their PPI networks, which they inherited from a common ancestor. Our results show that finding that overlap is not as easy as was previously thought. We can either align two networks in a way that matches many neighbors in $G_1$ to many neighbors in $G_2$, or we can align two networks in such a way that proteins with similar GO annotations are matched together. It does not appear from our results that it is possible to do both well. There are a number of possible reasons for this that should be better explored in future work. First, as we noted in Chapter 3, this tradeoff does not exist in synthetic networks. It’s possible that currently known protein networks have too many false positive and false negative edges to be aligned reliably, and current aligners,
ours included, waste time trying to make sure that false positive edges are conserved. If this is the case, we should probably prefer alignments with higher GOC, even if they have a lower $S^3$. Second, it is possible that our current best topological metrics are insufficient and mislead Optnetalign. These metrics all essentially count how many neighbors in $G_1$ have been matched to neighbors in $G_2$, but they have no way of ensuring neighbors of neighbors are matched to neighbors of neighbors, or that other large structures between networks are matched properly. One proposed metric that partially overcomes this is to report the size the largest connected common subgraph, but we have found that this metric correlates so strongly with metrics like $S^3$ that it does not provide us with much additional helpful information. Furthermore, this metric is not fine-grained enough. Conserving one extra edge could join two previously disconnected common subgraphs and essentially double the score. Equivalently, reporting the size of the largest common connected subgraph doesn’t tell us anything about the distribution of the sizes of the remaining connected subgraphs found. A few moderately large ones may be much more helpful than one very large one, especially if the species are more distantly related. For an illustration of these concepts, see Figure 6.1. Our algorithm performs well even when optimizing many objectives in one run, and, as we show in the previous chapter, this can be used to critically evaluate and compare different candidate objectives.

6.2 Similarity Estimation Functions

It would also be possible to use Optnetalign to evaluate existing aligners’ approximate topological similarity metrics, such as GHOST’s [74] spectral graph signatures, or the graphlet degree signatures used by the GRAAL family of aligners [78], just as we evaluated $EC$, $ICS$, $S^3$, $LCSC$, and $GOC$ in Chapter 5. The relative performance of these simi-
Figure 6.1: An example of the problems with measuring the number of nodes or edges in the largest common connected subgraph (LCCS) found by an alignment. Bold edges denote edges conserved by the alignment. Note that the LCCS consists of 5 edges, while the second largest contains 4. If the dashed edge were conserved as well, the number of edges in the LCCS would jump to 10. More nuanced ways of characterizing such situations are needed.

Similarity estimation functions has always been unclear, since they are always paired with a novel matching algorithm as well. With metaheuristic multiobjective alignment, we could optimize many of these metrics at once and determine which ones best predict or conflict with the objectives of network alignment. It’s even possible that some of these similarity measures, when paired with Optnetalign, could produce better results than optimizing $S^3$ or $GOC$ directly.

While our experiments in Chapter 4 on using machine learning with synthetic network alignment performed poorly compared to most existing alignment algorithms, they did, however, perform dramatically better than random alignments, as well as better than some older aligners. It may be interesting to combine machine learning methods with our new
aligner to learn what features between two networks cause our aligner to align certain nodes
to other nodes. This would be even easier since our aligner outputs so many alignments
in one run. Additionally, our preliminary work in Chapter 4 should be repeated with real-
world datasets, but instead of classifying whether a node is correctly aligned or not (which
is unknown with real data), we could instead use machine learning to predict the extent to
which aligning a given pair would contribute to our alignment objectives of $S^3$ and $GOC$.

The downsides of using similarity estimation functions, though, are that they are
expensive to compute and inherently imprecise. Given these downsides, are similarity es-
timation functions still useful when our aligner can produce such high quality alignments
without them? This is another research question for which we do not yet have an answer.
It has been shown [83] that MAGNA, a genetic algorithm for alignment, can produce better
results when it is seeded with the output of existing aligners that use similarity estimation
functions. The same may be true of Optnetalign, but it would certainly be more efficient
to build the seeding step into the program instead of relying on a third-party program,
especially given how prone to crashing most alignment algorithms are. A good course of
action would be to first verify whether seeding helps, and then determine which aligners
produce the best seeds. After determining that, the next step would be to use the design
of those aligners as a basis for exploring other similarity functions that could be built into
Optnetalign and refined with machine learning techniques.

6.3 Combining and Summarizing Alignments

Another possible direction for future work is to find ways to summarize or condense
the large number of possible alignments between two PPI networks. The original goal of
pairwise global alignment was to produce the one best alignment given our objectives. Our
work here has shown that there is no such alignment. Instead, there are many diverse alignments along a wide Pareto front, and these alignments may agree on as few as 1% of their aligned pairs while still appearing good with respect to at least one objective. An obvious next step would be to find ways to summarize the results of these alignments so that someone who wants to use an alignment can either pick the alignment that best fits their particular needs, or produce a composite alignment that summarizes the most interesting alignments found. We have done some preliminary experiments with creating *partial* alignments, using the number of aligned pairs of the alignment as an objective. This could be used to produce small, local alignments and large, global alignments in one run of the program as well, which would unify the two related, but heretofore separate, techniques of local and global network alignment. There are many other possibilities for producing summary or approximate alignments, as well. The recent publication of DualAligner [85] is a good step in this direction, since DualAligner precisely matches high-confidence matching proteins, but only performs coarse region-to-region alignment for low-confidence areas. However, this approach does not take into account hundreds of initial global alignments, as approaches based on Optnetalign could.

Graph querying techniques could also be adapted to understanding large amounts of alignment data. For instance, a tool could be developed that would allow the user to specify a given protein pathway, and query which sets of proteins the pathway was mapped to in another species. In some cases, such as the alignments with higher topological score, it could be that several pathways of similar shape, if not similar function, have been found in the other species, a situation analogous to that of many local aligners. In cases where biological function has been better matched, it may be that this pathway has been mapped to disconnected nodes in the second species that *should* be connected, but the interaction
data does not yet exist. Or, if no equivalent pathway exists in the second species, we may yet see some subtle similarity that sheds light on how the pathway evolved. The problem of graph querying has been extensively studied in the context of large biological networks. One paper [52], which studies the problem of efficiently extracting shared subgraphs from networks with common node labels, could be applied to alignments easily. PathBLAST [44] is a local alignment algorithm for network querying that is primarily used to identify conserved pathways. SAGA [91] is another approximate graph querying program for biological networks that scales to large graphs using an efficient indexing technique. CNetQ [40] is a biological network querying library for R based on conditional random fields [59], and has been adapted to perform network alignment as well. None of these querying systems have been adapted to alignment-based querying, but this would be very feasible – the queries could be run on either $G_1$ or $G_2$ individually, or could be run on the connected components shared between them, as found by Optnetalign.

The main challenge in producing summaries of many alignments is making topological sense of the summaries. We can easily report, for instance, that aak-2 in *C. elegans* is aligned to PRKAA2 in *H. sapiens* in 95% of output alignments, but we can’t as easily summarize how consistently or how often the subnetwork around aak-2 is mapped to the subnetwork around PRKAA2. As can be seen by casual use of bioinformatics network visualization software such as Cytoscape [89], it can be very difficult to visualize a single PPI network, since they are so large that most visualization algorithms produce uselessly messy visualizations, known informally as “hairballs”. It seems that it would be much more difficult to visualize an imperfect overlay of two of these networks. We also can’t easily report rare, exceptional results that may be of biological interest. It may be that all of the remaining alternatives aak-2 is mapped to are obviously wrong, except for one that
gives us some unexpected insight. It would be difficult to prevent such interesting minority information from being buried in the large volume of data that is produced by network alignment.

Several recent visualization methods for large networks could be adapted to help visualize alignments. Hive plots sort nodes on axes by topological metrics instead of laying them out arbitrarily [55]. BioFabric [63] displays nodes as horizontal lines, and edges as vertical lines, so edges do not overlap and large graphs can be more easily inspected. These techniques have been used for visually inspecting single graphs, instead of alignments between graphs, but by combining two networks and using labeled edges, we could easily make an initial visualization prototype. Adapting these methods or developing new methods for visualization would be extremely useful in making network alignment a mainstream bioinformatics tool – while sequence alignment tools have easy-to-read textual representations of alignments, accomplishing the same for network data is much more difficult.

6.4 Large-scale Network Analysis

In line with our overall desire to see PPI network alignment shift its focus from producing alignment algorithms to producing alignments themselves, we believe it is time to begin experimenting with aligning protein networks at a large scale. Databases are now available that contain protein interaction information for hundreds of species, such as STRING [31], which contains 1,133 species. A stripped-down version of Optnetalign that only performs hill climbing can produce competitive alignment results in as little as 30 seconds. With enough processors working in parallel, it would be possible to align all of the species of STRING in a few days. This would provide a wealth of interesting data that could be analyzed in a number of novel ways. We expect that our next project will be to
attempt to verify that topological alignment can be used as a similarity measure between species that can reproduce phylogenetic relationships [57]. Existing alignment algorithms have never been tested on more than ten species at a time, so increasing that number by two orders of magnitude by using STRING would be a novel endeavor.

Kuchaiev et al. [57] present some evidence that, by aligning metabolic networks of different species, they are able to recreate phylogenetic trees using an average distance method, where distances are calculated using EC. This was the only information they used to create these trees, and it appears to give us evidence that information on the topology of metabolic networks, which seems entirely orthogonal to the standard distance measure of sequence alignment, can be used to create the same trees sequence alignment data does. It would be interesting to attempt to reproduce these results using protein networks. There are many existing R packages [84] for phylogenetic tree inference, and we could make use of these to produce the trees.

Another novel application would be to use a big data approach to perform multiple alignment, by mining the resulting alignments for transitive alignment pairs. For example, if protein $u$ in species $x$ has been mapped to protein $v$ in species $y$, and $v$ in turn has been mapped to protein $w$ in species $z$, we could determine whether $u$ has been mapped to $w$ as well. Multiple alignment algorithms, which align many networks simultaneously, require alignments to be transitive in this way, but since pairwise aligners such as Optnetalign only consider two networks in isolation when aligning, transitivity is not maintained across larger sets of protein networks. Examples of multiple alignment algorithms include Graemlin 2 [29], IsoRankN [61], NetworkBLAST-M [43], Shih and Parthasarathy’s unnamed aligner [87], and SMETANA [82]. These aligners work by outputting equivalence classes of proteins that should be mapped to one another across all $k$ protein networks that have been provided as
input. The most recent of these algorithms are somewhat scalable, with polynomial time complexity with respect to $k$, and SMETANA has been applied to values of $k$ as large as 20. However, for a large database of over a thousand species, multiple alignment would be infeasible using existing algorithms. The STRING database is hundreds of gigabytes in size, and existing multiple alignment algorithms all assume the data set can fit in memory. However, by using Hadoop and our pairwise alignments, it would be possible to create multiple alignments. This would allow us to process the large amounts of pairwise alignment data across many machines, and use that data to perform multiple alignment in a distributed fashion.

Several methods of producing multiple alignments from many pairwise alignments are possible. Here, we explain two simple ones. The simplest option would be to simply append pairwise alignments. Say we have $k$ species. We would first choose the highest-quality pairwise alignment from among them as our seed. This would account for the first two species of the multiple alignment. Then, to add additional species, we would take the highest-scoring alignment between one of the two species already included in the multiple alignment and one of the species not yet included in the alignment and add it to the multiple alignment, so that the multiple alignment then contains three species. We would repeat this process until all $k$ species have been added to the alignment. This would essentially be a greedy approach to multiple alignment, and this could be further improved by randomization of the pairwise alignments used. Then we could generate many multiple alignments, evaluate them, and take the best. A second means of producing multiple alignments, that takes a map-reduce approach, would be to use an existing multiple aligner such as SMETANA, or even one of our own design, and use it as a map step, performing multiple alignments of, say, 20 networks in each job. Then, the reduction step would be to “stitch together” the small
multiple alignments using the highest-scoring pairwise alignments we have already created. By varying how many networks to use in the individual multiple alignment jobs, we would effectively be adjusting the tradeoff between computational efficiency and accuracy. See Figure 6.2 for details.
When given $k$ networks, each of the $p$ jobs is given $k/p$ networks and produces an initial multiple alignment between them (step 1). These alignments consist of a set of equivalence classes. For instance, the $u$th equivalence class identifies nodes from networks $G_1, G_2, ... G_{k/p}$. To combine the output of two jobs, we find the best pairwise alignment (previously computed) between the networks of the two small multiple alignments. The equivalence classes are then combined into larger ones using the pairwise alignments as a guide. This is a simple way to leverage our fast pairwise aligner to scale up the capabilities of conventional multiple aligners to large data sets.

Figure 6.2: Schematic of the proposed process to combine small multiple alignments into a large one.
6.5 Conclusion

Let us summarize what we have accomplished in this thesis. We have first reviewed the problem of PPI network alignment and its purpose, showing that several distinct formulations of the problem are possible. We decided to focus on global pairwise network alignment. We then surveyed the most successful global pairwise alignment algorithms and classified them according to common design decisions, especially taking notice that most aligners follow a two-stage approximate alignment process that measures alignment success after the alignment is completed. We then performed the first comprehensive benchmark comparing many of these aligners to one another, and found that they differ greatly in performance, and in which objective they better optimize.

The latter part of the thesis focused on overcoming the deficiencies of the aligners we had previously surveyed. We found reasons to deviate from the common two-stage alignment paradigm, and, since joint optimization of both alignment objectives was a problem with existing aligners, we focused on multiobjective algorithms. First, we investigated seeding our metaheuristic aligner with approximate matchings using a machine learning approach. However, this was quite slow and we found that it could not generalize well to real-world networks. Leaving machine learning approaches for a future research project, we proceeded to introduce our multiobjective memetic algorithm, Optnetalign, which outperformed existing aligners in most cases. We used Optnetalign to better analyze the differences and similarities between different alignment objectives and to better characterize the problem of network alignment, showing that the common assumption that one best alignment can be found is incorrect for current data sets. Instead, a huge number of potential alignments can be created at different points in objective space. The speed and multiobjective nature of our algorithm open the door to many potential research projects in the future.
REFERENCES


