Global Alignment of Protein-Protein Interaction Networks Using Subgraph and Sequence Similarity

Camille C. Diez
Ateneo de Manila University
Katipunan Avenue
Quezon City, Manila 1108
diezcami@gmail.com

Enrico M. Tria
Ateneo de Manila University
Katipunan Avenue
Quezon City, Manila 1108
evmt15@yahoo.com.ph

John Paul C. Vergara
Ateneo de Manila University
Katipunan Avenue
Quezon City, Manila 1108
jpvergara@ateneo.edu

ABSTRACT

This paper provides a model that accepts two protein-protein interaction networks and displays a mapping of their maximum common sub-network and information regarding the alignment. The core of this model is an algorithm that uses sequence similarity and max cliques to globally align and map the maximum common subgraph between protein-protein interactions modeled as networks or graphs. The aligning of these networks has many applications in biological research and theoretical computer science, specifically in the fields of graph theory, computational science and algorithm design. To achieve this, the researchers discuss the methods, limitations and recommendations of prior works and algorithms to derive aspects from which the algorithm can be improved. The researchers then formulate the algorithm and assess its performance by computing the edge correctness and obtaining the percentage of the graph that has been accurately mapped. Finally, the algorithm is integrated into the model and an overview of its performance is presented, focusing on its accuracy and efficiency.

CCS CONCEPTS

•Theory of computation → Sparsification and spanners;

KEYWORDS

PCSC proceedings, Protein-Protein Interaction Networks, Sequence Similarity, Maximum Subgraph, Global Alignment

ACM Reference format:
Camille C. Diez, Enrico M. Tria, and John Paul C. Vergara. 2017. Global Alignment of Protein-Protein Interaction Networks Using Subgraph and Sequence Similarity. In *Proceedings of 17th PCSC, Philippines, March 2017,* 12 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Context of the Study

The study of protein interactions is a key fixture in the field of biology, given how they play an important role in explaining the various functions and components of cells. One interesting subtopic under this study is that of interactions between proteins, referred to as protein-protein interactions or PPIs. These interactions involving multiple proteins refer to physical contacts created as a result of biochemical events or other forces [16]. In the field of molecular biology, these interactions can be represented through species-specific interactomes, which can then be mapped into graphs. Advances in technologies have enabled rapid large scale data generation of PPI networks at the species level, which paves way for the comparison of proteins and functions across species. Both sequence (tied to the molecular level) and topological (interactions conserved across networks) similarities have to be taken into account to match proteins across networks. Sequence-based comparisons have been the workhorse of bioinformatics for the past four decades, furthering our understanding of gene function and evolution [11].

Outside the field of biology, PPI networks are also relevant in the field of computer science as the problem of network alignment can be approximated using graph theory. PPI networks can be mapped into standard graphs with nodes representing proteins, edges representing interactions, and the whole structure representing a network which can be tested for alignment with other graphs. The recurring problem of PPI network alignment using sequence and topological similarities as conditions can be analyzed using graph analysis techniques, specifically those used to approximate the theoretical problems of graph isomorphism and finding the maximum common subgraph (MCS) between any given graphs.

Aside from graph theory, this study uses methodologies developed in several other subfields of computer science such as data analysis, computational science and algorithm design. Data analysis will be heavily used to compare and inspect the various graph examples involved while carrying out the research. Computational science refers to coming up with solutions to scientific problems by means of using the principles of computer science, with the problem in this particular context defined as matching various PPI networks. Algorithm design is also used heavily in this study in the context of researching, implementing and creating algorithms to approximate the defined problem.

1.2 Research Objectives

Given the context of the study, the core aspect of the PPI network problem this study aims to work towards is creating an accurate and efficient way to identify similarities in the PPI network structures of organisms across different species.

As an approximation to the core problem stated earlier, this study aims to develop a valid model for displaying the maximum common subgraph between two given protein-protein interaction networks. Prior to building the model, an algorithm will be formulated to determine the largest common subgraph between two PPI network graphs.

The maximum common subgraph problem in question is NP-complete, meaning the solutions to these problems can only be found in polynomial time by means of a non-deterministic algorithm capable of guessing the answer correctly at each step [5]. The approach to be used in the paper involves utilizing a greedy heuristic to approximate the maximum common subgraph of two graphs instead of an algorithm that determines the maximum common subgraph between two networks. As such, the study contains two sub goals that work towards the implementation of this model:

1.2.1 Formulating the Algorithm. An MCS algorithm will be created to approximate the problem of finding the largest common subgraph given two graphs. This process consists of researching the contexts, strengths and weaknesses of similar algorithms, as well as testing and assessing the created algorithms given a wide variation of sample graphs. In deriving the algorithm, we plan to use the principles of max cliques to find the most suitable comparison between the two graphs, as well as sequence similarity to create heuristics and break ties in the algorithm. Sequence similarity will provide a numerical measurement of similarity between the vertices of separate graphs, which in turn allows the algorithm to better determine not only similarities, but isomorphism between graphs in the case of ties.

1.2.2 Developing the System. After reaching a sufficient level of efficiency and accuracy from assessing created algorithm, a system will then be developed with the algorithm at its core. The system will accept two PPI networks mapped into graphs and output the mapping of their maximum common subgraph. The output will consist of the vertices and edges mapped for both graphs and several details regarding the mapping, including but not limited to the number of vertices mapped and the percentages of each graph mapped to each other.

1.3 Research Questions

Aside from being able to approximate the solution to core problem defined earlier, the model produced in this study is also expected to have a good level of accuracy and efficiency in its implementation. It was stated earlier that the performance of the algorithm would be improved using heuristics from applying several principles of sequence similarity and graph isomorphism. As such, this study aims to answer the question - *How can sequence similarity and graph isomorphism be utilised to derive an efficient MCS algorithm*? In answering this, several sub-questions come to mind:

(1) Aside from sequence similarity, what properties of graphs can be utilised to break ties and determine the most accurate levels of similarity between PPI networks?
(2) How can graph isomorphism be used to further improve the accuracy and efficiency of gauging the heuristics of the algorithm?
(3) In light of theoretical computer science, what aspects of PPI networks can be utilized in order to narrow down the scope of approximating the NP-complete largest common subgraph problem?

1.4 Scope and Limitations

The scope and viability of this study is greatly influenced by the availability of large PPI network datasets provided by STRING [23], NAPAbench [17] and IsoBase [13]- all of which allowing us to present accurate results and analyses of our research. As such, the basis of the sample data, algorithm assessments and overall analyses throughout the research will be dependent on the data made available from these sources.

As the PPI network alignment problem is NP-Hard, deriving an efficient and complete solution for it is difficult and likely impractical given the time constraints in completing this research. Complete PPI networks average out to having at least 4000 nodes per graph, greatly affecting the performance of the algorithm, as well as the time taken to completely test and analyze the results. Most datasets containing PPI networks are also non-synthetic, and thus don't contain expected solutions that can be used to evaluate the correctness of the algorithm.

In line with this, tests for the algorithm have been limited to the generated datasets presented in this paper. The graphs in these datasets replicate the characteristics of PPI networks (i.e., The presence of sequence similarity and topology) while having significantly smaller graph sizes, and also have expected solutions used in testing the correctness of the algorithm. As such, the algorithm presented in this thesis will likely only be able to identify the maximum common subgraphs between subgraphs of networks.

This study also focuses on approximating the problem of global network alignment in the context of PPI networks. As such, while the algorithm formulated may be beneficial in its formulation to approximating the theoretical problem of finding the MCS between graphs, it works best only within the context it was created for.

1.5 Significance of Study

In addition to the possibility of becoming a subject of further research for PPI networks, approximating the problem of globally aligning PPI networks reaches out to members of the scientific community as well. The findings of this research is significant to both the field of theoretical computer science as a whole and to the concrete research projects in the field of biology.

In the field of biology, approximating the network alignment problem paves way for researchers in the field of molecular biology to derive further insight into the role of proteins, which in turn provides further understanding to biology as a whole. The aligning of PPI networks also has many applications in general biological research [7].

In the field of theoretical computer science, formulating a solution to the core problem of this study provides additional insight into the general network alignment problem. Furthermore, the development of an accurate and efficient algorithm for the alignment of these networks presents a scalable solution, as it is extensible not only to local network alignments, but to multiple network alignments as well. In developing a solution to PPI network alignment, solutions to problems in theoretical computer science are also being developed through the study of methodologies and factors mandatory for creating accurate and efficient algorithms. By creating an algorithm that approximates the alignment of these PPI networks, insights can be created and further applied to the global network alignment problem.

Given the global importance and prevalence of computer science and biology, it can be inferred that the benefits of exploring PPI network interactions will reach the general public regardless of their involvement in either field. The developments in biology will provide further understanding to relationships between species including our own, as well as advancements in the field of medicine from gaining a better understanding of PPIs. Further insights into approximating the problem of global network alignment in the context of theoretical computer science allow for developments in approximating the P vs. NP problem, which opens up solutions to many tasks and problems people face today.

2 DEFINITION OF TERMS

2.1 Protein-Protein Interactions

The sub-field of protein-protein interaction networks is vital in the field of Biology. They can be used to understand the molecular makeup of organisms and diseases. Because of the sequences and connections - interaction - between proteins, it leads to the identification of even unknown functions [19]. While the interactomes of organisms do not appear to be structured in a similar manner, they share common interactions internally. As such, the solution to finding the common structures in differing PPI networks are closely aligned to the solutions to finding the the largest common subgraph between graphs - a known problem in the field of theoretical computer science. In addition to this, PPI networks are crucial for all biological processes wherein the interactions can give insights to protein function [22].

These networks can become very complex because of the large number of proteins in an interaction network. Network science deals with this by simplifying the networks into components which will serve as nodes or vertices and interactions which will serve as the edges. The nodes in a network representation are metabolites or macromolecules. Metabolites are the products of metabolism while macromolecules are very large molecules that are composed of usually thousands of atoms. Examples of these are proteins, RNA molecules and gene sequences, while the edges are physical, biochemical or functional interactions [19].

2.2 Sequence Similarity

Detecting similar pairs in large biological sequence collections is one of the most commonly performed tasks in computational biology [24]. As these datasets can grow very large in size, it gets difficult for algorithms to determine exact matching common subgraphs as the mappings become ubiquitous. A large graph may have numerous subgraphs that are identical with each other in terms of topography and structure. This may lead to confusion in choosing the matching subgraph. In these situations, sequence similarity can be incorporated in algorithms to provide a better context to the mapping of vertices. Even if a graph has multiple identically structured subgraphs that can possibly be detected for matching, the one that has a higher sequence similarity score will be chosen. The functions of sequence similarity are mostly used when comparing two different graphs, as it provides further information regarding their relationship. This information concerns measures of similarity of a vertex from the first graph to every other vertex of the second graph, and applies to all vertices in each graph. The level of similarity can be gauged by different factors such as the number of edges, positioning, or further qualities of what the graph represents in the real world.

The similarity between vertices is measured numerically in order for it to be used in further analysis and algorithms. The ranking system for sequence similarity works both ways - high values may either represent a high lever of similarity between vertices or little to no similarities at all, depending on the implementation and problem being solved. In the context of the MCS problem, sequence similarity can be used to predict and select the matching vertices from two graphs that can potentially be a part of their largest common subgraph.

Using sequence similarities to map networks in the context of biological research is not unheard of - Matsuda has conducted a study classifying molecular sequences with their pairwise similarities [10], and presented a method for classifying a large and mixed set of uncharacterized sequences. Aside from just using graph properties, sequence similarity scores can also be determined through dynamic programming algorithms, such as the Smith-Waterman local alignment algorithm [10].

2.3 Cliques

A clique is a complete subgraph of a given graph [3][15]. This is a type of subgraph wherein all the vertices are interconnected with each other. A clique is said to be maximal if it is not properly contained in another clique of the given graph [15]. A maximum clique, on the other hand, is the biggest clique in the graph.

Previous works have proposed that a maximum clique algorithm can be used to solve graph matching problems or the maximum common subgraph problem. This is done by taking two separate graphs and then creating an association graph based on the two original graphs. The size of the association graph is the size of both original graphs multiplied together. Each vertex in the new graphs corresponds to a combination of one vertex each from the two original graphs. An edge is created in the association graph if there is a correspondence between the vertices in the two original graphs [4].

There has been previous algorithms using the same principles. A notable algorithm is the Durand-Pasari algorithm. The first step in this algorithm is the construction of the association matrix. The edges represent the compatibility of the various pairs of vertices from the two starting graphs. These edges are undirected. For example, a node corresponding to a pair (n1, n2) will have a connection to a node corresponding to (m1, m2) if and only if there is an isomorphism between the subgraph n1, m1 of the first graph and the subgraph n2, m2 of the second graph. This condition can be checked by looking at the edges between n1 and m1 and between n2 and m2 in the two starting graphs while also taking into account node and edge attributes [6]. This would show that those nodes exhibit the same relationship: either both adjacent or both non-adjacent [8].

The next step of the algorithm is the clique detection itself. It uses a depth-first search strategy on a search tree by selecting one vertex at a time from different levels until there are no more vertices that can be added to the list [9]. This will then yield the maximum clique, which corresponds to the maximum common subgraph of the two initial graphs.

2.4 Variations in Network Alignment

Network alignment is the problem of finding similarities between the structure or topology of two or more networks [12]. The main goal of biological network alignment is to help identify functional orthologs across different species [2]. There are many variations of the network alignment problem, namely the local and global network alignment problems. The goal in local network alignment (LNA) is to find small, local regions of isomorphism between the input networks of two given graphs, with each region implying a mapping independently of others [21][7]. Due to this, each node may be mapped to numerous pairings, resulting in local alignments that are not mutually consistent and therefore ambiguous. Older works on PPI network alignment almost exclusively focused on this problem. This is problematic as ubiquitous networks can affect the performance and accuracy of algorithms used on PPI networks, especially given their sizes. Some arguments have been made that LNA is more faithful to biological theory, by explicitly acknowledging the possibility that a protein can have multiple orthologs; however, this is difficult to interpret in the context of computer science, and has therefore died down in popularity [7].

On the other hand, the goal in global network alignment (GNA) is to find the best overall alignment for each node between the two input networks by defining a single mapping across all parts of the input graphs [21]. Its alignments are mutually consistent as a result of the comprehensive mapping - that each node in an input network is either matched to some node in the other network or explicitly marked as a gap node (i.e., with no match in the other network) - for each global alignment [21]. As stated by Clark *et al.*, the similarity between two nodes in different networks can take the form of topological similarity between the neighborhoods of the respective nodes in their graphs, or biological similarity, often measured by sequence similarity [7].

This study focuses on solving the GNA problem in the context of PPI networks by taking advantage of sequence similarities between the given graphs. This is due to the GNA being a viable solution to detecting related maximal sub-networks between graphs [7]. ‘

3 METHODOLOGY

3.1 Methodology as Replicable

The core research of this study lies in formulating the algorithm that would be used in the model it produces. As such, several variables are defined to make the process systematic and extensible to factors outside the scope of this paper.

In the process of formulating the algorithm, the independent variables are the aspects of the algorithm modified in order to determine the best specification mix for the PPI network alignment problem. These aspects mainly revolve around how the heuristics are formed via sequence similarity and graph isomorphism. The dependent variables of the study are the different evaluations yielded by the modified versions of the algorithm. These different evaluations will serve as the resulting variables that will be compared and assessed throughout the course of this study. Finally, in validating and assessing the performance of the algorithm, the controlled variables are the the PPI network inputs that must remain the same for every set of tests.

3.2 Methodology as Realistic and Time-Bound

The methodology is made to be realistic as seen in the results of the related studies that have been accomplished in the past, such as the Exploration of Protein-Protein Interaction Mappings [18] done in 2015 in the Ateneo de Manila University in under a year. Algorithms such as Spinal [1] and PINALOG [14] have also been formulated, developing improvements to the previous methods to solve the PPI network problem. As many of the research components including the preliminary reviews, formulating the algorithm, and assessing its results will borrow techniques from these prior works, it can be concluded that the methodology is realistic as it is the output of many concrete and replicable studies.

The methodology can be accomplished given a time frame of 6-7 months, with its main components being the MCS algorithm to be created, the algorithm for generating datasets, the model for evaluating PPI networks and the assessment of several evaluations and results yielded by the study.

3.3 Summary

Before approximating the core parts of the study, researching on prior work and related algorithms was done in order to provide a clearer context to the limitations and possible avenues for improvement for the study. In addition to this, research on the subfields of theoretical computer science, particularly graph theory, has been done to get a better understanding of the fundamental concepts to be used in the study.

After adequate research, the preliminary work for the formulation of the algorithm was done. As an immediate goal, we formulated small samples of graphs, with approximately six to ten vertices each. These graphs were be tested on using a MCS algorithm that initially accommodates smaller sample inputs. Each input graph had have its own matrix containing further information on the edges of the graph such as the scaled actual distances and lengths between two vertices. In the event of ties - which occur when there are multiple subgraphs in one graph common to that of the other graph - the aforementioned matrices were be used within the context of finding the sequence similarity between the given graphs to determine which subgraph among the multiple located ones to use. The matrices also serve as guides to determine the vertices and edges to be considered in building the subgraphs. Throughout this stage, principles of computational science were used to aid the flow and development of the research.

After the preliminary work, the initial algorithm was extended to be able to handle large PPI networks. Principles of sequence similarity and graph isomorphism will be incorporated to improve the overall performance of the algorithm. Several tweaks were be done on the method of getting heuristics in order to determine the variables that best suit accurately assessing the global alignment of PPI networks. Evaluations and assessments were periodically be conducted in order to systematically determine the best solution to the main problem of the study. Principles of algorithm design were heavily be used throughout this stage of the methodology.

After the formulation of the algorithm, a valid model accepting inputs of PPI networks and outputting results of their alignment was developed and assessed. Throughout the bulk of the methodology, the datasets will be generated using an algorithm, with the format based on the [17] datasets. Cytoscape, an open source platform for complex network analysis, will also be used to display the visual mapping of the subgraphs between the PPI networks, as well as parsing the datasets provided by the aforementioned databases [20].

3.4 Algorithm Variants

We created three variants of the MCS algorithm, each with substantial improvements over the other in detecting the MCS between two graphs. All three are similar but there are progressions between variants.

The general structure of each algorithm is that of a Maximum Clique algorithm. The Maximum Clique Algorithm used is based on the other Max Clique algorithms mentioned in the related literature. It is comprised of two steps, namely creating the association or correspondence graph and finding the maximum clique of the aforementioned association graph.

The proposed algorithm does not make use of an actual association graph in order to save space. Association graphs tend to become very large since it is a two dimensional array that is the size of the two original graphs multiplied together. For example graph 1 is of size $n1$, and graph 2 size $n2$. The size of the association graph would be $n1^2$ x $n2^2$. In order to avoid this, a mathematical function will be used to get the value of a certain vertex in the association graph if there were an actual association graph.

Although there isn't an actual association graph, the algorithm makes use of a function that computes for the index in the association graph based on indices from the two original graphs. It is a simple function that is important so that the algorithm can easily transition in viewing the problem from the two initial graphs to the association graph.

The part of the algorithm that finds the maximum clique begins with scanning through all the vertices in the association graph. These vertices will then be put in a priority queue *Maximum* based on certain heuristics. This is the priority queue which checks for the vertices with the maximum score based on the used heuristics. These heuristics differ for all 3 algorithm variants. These heuristics include weighted edges and sequence similarity scores. This priority queue *Maximum* will be useful as the algorithm progresses.

The concept of weighted edges is used to determine the importance of the certain edges in the association graph. As mentioned in the RRL, edges in the association graph are made if there is a similarity between the certain vertices between the two initial graphs: either there are both actual connections or both non-connections. Weighted edges put more weight on the actual connections.

Sequence similarity scores are fed into the algorithm via input. These scores are based on the actual biological structure of PPI networks. The higher the score, the more similar two vertices are.

From the initial priority queue, the top ten vertices in the will be checked for a maximum clique. This means that the algorithm runs ten times. Each vertex in the top ten will serve as a starting point for each time the algorithm runs. From that starting point, the algorithm will look for a clique. Since there are different starting points, ten different cliques will be detected. At the end of the algorithm, the maximum will be selected from those ten cliques. This algorithm that runs ten times compared to the exact algorithms that check all of the vertices performs quicker.

For each of the ten runs of the algorithm, the process goes as follows. After selecting a top vertex Top that will serve as the starting point from the initial priority queue *Maximum*, the algorithm will check all of its adjacent vertices. These adjacent vertices are placed in another priority queue $Neighbors$ where they are arranged by their number of edges and number of edges connected to the same neighborhood. By neighborhood, it means the vertices that are connected to the top vertex. The priority queue $Neighbors$ would then contain all of the adjacent vertices of the starting point Top.

After building the priority queue $Neighbors$, detecting the clique comes next. There is a list $Clique$ that tracks the current clique being formed. To start off, Top is added to $Clique$. Afterwards, the algorithm will continuously poll the priority queue $Neighbors$. For each polled vertex, the algorithm will check if that vertex has a connection or an edge between it and every vertex in the list $Clique$. If it does, the algorithm will just continue on polling. If it does not have a connection, the algorithm stops and whatever is in the list $Clique$ is considered the clique found for that certain run of the algorithm.

This process goes on for ten runs. For each run, a clique is formed. After ten runs, the algorithm would have detected ten cliques. The biggest clique out of the ten will be chosen as the maximum clique. The vertices in the maximum clique will then be decoded back to its corresponding vertices in the two original graphs to represent the most common subgraph between them.

3.4.1 *Variant 1: Weighted Edges.*

For this variant, the heuristic used is the weighted edges of each vertex. It also makes use of the connection to the same neighborhood, but all of the variants make use of that as well.

These heuristics are used in the two priority queues in the algorithm: *Maximum* and $Neighbors$. Because of this heuristic, this certain variant prioritizes vertices with more adjacent vertices. Although the scores that are being checked focus on the weighted edges, a high score still means that the vertex has many adjacent vertices. This is because the score of the weighted connection is 2 while that for a regular connection is 1. It is also good that the algorithm works with vertices with many adjacent vertices. This gives the algorithm more chances to expand the clique that is being formed because of the abundance of said adjacent vertices.

To hold the weighted edges value, the algorithm deploys an array called $Edges$. The size of this array is the length of the association graph. The algorithm can then easily retrieve the weighted edges values of certain vertices when needed during run time.

3.4.2 *Variant 2: Tie-Breaking Using Sequence Similarity.*

For the second variant, the heuristics used are weighted edges and sequence similarity scores. The sequence similarity scores are added to the weighted edges of each node to form the numerical value that is checked in the priority queue. The higher this value, the more priority is given to that node.

The addition of sequence similarity scores to the weighted edges help in ensuring that some importance is given to other characteristics of the vertices. In the PPI context, these characteristics

Algorithm 1: Maximum Clique Algorithm used for the MCS problem

```
1 function MCS ($n1, n2, L1, L2$);
  Input  : The number of vertices in two graphs $n1$ and $n2$, and
           their edge lists $L1$ and $L2$
  Output : The MCS between the input graphs
    $Graph1$ ← Equivalent adjacency matrix of $L1$
    $Graph2$ ← Equivalent adjacency matrix of $L2$
    $Edges$ ← Array of length ($n1$ * $n2$)
    for all "Vertices in theoretical correspondence graph ($Graph1$,
    $Graph2$) of length ($n1$ * $n2$)" do
      $Current$ ← Current vertex
      for all "Neighboring vertices of $Current$" do
        Add total edges of $Current$ to $Edges$ at index $Current$
      end for
      Add $Current$ to Priority Queue $Maximum$
      $Maximum$ checks for Weighted Edges
    end for
    $MaxClique$ ← Null
    $CurrentClique$ ← Null
    for all "Top 10 Vertices in $Maximum$" do
      $TopVertex$ ← Head of $Maximum$
      for all "Adjacent Vertices of $TopVertex$" do
        $TempVertex$ ← Current adjacent vertex
        for all "Adjacent Vertices of $TempVertex$" do
          Count number of Vertices that are connected to
          $TopVertex$
          Total count will serve as the $ConnectiontoNeighborhood$
          value of $TempVertex$
        end for
        Add $TempVertex$ to Priority Queue $Neighbors$
        $Neighbors$ checks the sum of Weighted Edges and
        Connection to Neighborhood
      end for
      Add $TopVertex$ to $CurrentClique$
      for all "Vertices in $Neighbors$" do
        $Vertex$ ← Current vertex
        if "$Vertex$ is adjacent to all vertices in $CurrentClique$"
        then
          Add $Vertex$ to $CurrentClique$
        end if
      end for
      if "$CurrentClique$ is greater than or equal to $MaxClique$"
      then
        $MaxClique$ ← $CurrentClique$
      end if
    end for
    Output all vertices in $MaxClique$
```

may be biological. This helps the algorithm in taking into account characteristics other than graph topology.

To accommodate the sequence similarity scores, an array *Sequence Similarity* is created to hold the sequence similarity scores of each vertex in the association graph. If the initial graph, graph 1 is size n and graph 2 is size m, the size of the sequence similarity array is n x m.

Algorithm 2: Maximum Clique Algorithm version 2

```
1 function MCS ($n1, n2, L1, L2, s$);
  Input  : The number of vertices in two graphs $n1$ and $n2$, their
           edge lists $L1$ and $L2$, and the list of sequence
           similarity scores of the association graph $s$
  Output : The MCS between the input graphs
    $Graph1$ ← Equivalent adjacency matrix of $L1$
    $Graph2$ ← Equivalent adjacency matrix of $L2$
    $Edges$ ← Array of length ($n1$ * $n2$)
    $SequenceSimilarity$ ← Array of length ($n1$ * $n2$)
    for all "Vertices in theoretical correspondence graph ($Graph1$,
    $Graph2$) of length ($n1$ * $n2$)" do
      $Current$ ← Current vertex
      for all "Neighboring vertices of $Current$" do
        Add total edges of $Current$ to $Edges$ at index $Current$
        Add sequence similarity scores from $s$ to
        $SequenceSimilarity$ at index $Current$
      end for
      Add $Current$ to Priority Queue $Maximum$
      $Maximum$ checks the sum of Weighted Edges and Sequence
      Similarity scores
    end for
    $MaxClique$ ← Null
    $CurrentClique$ ← Null
    for all "Top 10 Vertices in $Maximum$" do
      $TopVertex$ ← Head of $Maximum$
      for all "Adjacent Vertices of $TopVertex$" do
        $TempVertex$ ← Current adjacent vertex
        for all "Adjacent Vertices of $TempVertex$" do
          Count number of Vertices that are connected to
          $TopVertex$
          Total count will serve as the $ConnectiontoNeighborhood$
          value of $TempVertex$
        end for
        Add $TempVertex$ to Priority Queue $Neighbors$
        $Neighbors$ checks the sum of Weighted Edges, Sequence
        Similarity scores, and Connection to Neighborhood
      end for
      Add $TopVertex$ to $CurrentClique$
      for all "Vertices in $Neighbors$" do
        $Vertex$ ← Current vertex
        if "$Vertex$ is adjacent to all vertices in $CurrentClique$"
        then
          Add $Vertex$ to $CurrentClique$
        end if
      end for
      if "$CurrentClique$ is greater than or equal to $MaxClique$"
      then
        $MaxClique$ ← $CurrentClique$
      end if
    end for
    Output all vertices in $MaxClique$
```

3.4.3 *Variant 3: Full Implementation of Sequence Similarity.*

The third variant of the algorithm makes use of the same heuristics as the second one: weighted edges and sequence similarity scores. The difference between the two lies in how they are utilized in the checking of the priority queue. Instead of just adding the two values together, the sequence similarity scores are given more bearing than the weighted edges. In the priority queue, sequence similarity is checked first. If there are ties, then the weighted edges are checked.

In this variant, sequence similarity is given more importance than the weighted edges. This means that this variant focuses more on the biological characteristics of PPI networks rather than how the networks appear topologically.

Another difference for the third variant of the algorithm has to do with the clique that it outputs. The first two algorithms output the clique with the most number of vertices out of the top 10. In the 3rd variant, the important measure for the clique is the total sequence similarity score. Each vertex in the clique has a sequence similarity score, giving the clique itself a similarity score. A clique with a higher sequence similarity score is regarded as a better clique.

3.4.4 *Time Complexity.*

As the three algorithm variants follow the same processes, they run on the same time complexity as computed in the table below:

Table 1: Time Complexity

Module	Complexity
Assuming n	maximum($n1$, $n2$)
Creating two adjacency matrices	$2n^2$
Size of theoretical correspondence matrix	$n^2 * n^2 = n^4$
Counting number of edges of each vertex	n^4
* Adding to first priority queue	$log(n)$
Counting number of edges of top vertex	n^4
Adding to second priority queue	$log(n)$
Adding to the clique	n
Total	$n^4(log(n))$

3.5 Dataset Generation

In order to have better control over the graph datasets to be used in testing the algorithms, we formulated several algorithms to generate test cases. The datasets each consist of 25 test cases and are classified according to their characteristics and the presence of distraction graphs.

3.5.1 *Graph Generation.*

The graph generation parameter accepts the parameters G_c, n, e, d, and G_d for each of the two graphs to be generated for every dataset. G_c refers to the subgraph that will be common to two generated graphs, also know as the seed graph. n refers to the number of vertices that will be present in the graph, inclusive of G_c. e refers to the crossing edges that connect newly added vertices to G_c, while d refers to the inter edges that connect newly added vertices to each other.

The inter and crossing edges in both graphs will be generated randomly in order to distinguish G_c from the other edges, thus directing the algorithm towards identifying it as the largest common subgraph in each dataset. Since the edges outside G_c are randomly

Algorithm 3: Maximum Clique Algorithm version 3

```
1 function MCS ($n1, n2, L1, L2, s$);
  Input  : The number of vertices in two graphs $n1$ and $n2$, their
           edge lists $L1$ and $L2$, and the list of sequence
           similarity scores of the association graph $s$
  Output : The MCS between the input graphs
    $Graph1 \leftarrow$ Equivalent adjacency matrix of $L1$
    $Graph2 \leftarrow$ Equivalent adjacency matrix of $L2$
    $Edges \leftarrow$ Array of length ($n1$ * $n2$)
    $SequenceSimilarity \leftarrow$ Array of length ($n1$ * $n2$)
    for all "Vertices in theoretical correspondence graph ($Graph1$,
    $Graph2$) of length ($n1$ * $n2$)" do
        $Current \leftarrow$ Current vertex
        for all "Neighboring vertices of $Current$" do
            Add total edges of $Current$ to $Edges$ at index $Current$
            Add sequence similarity scores from $s$ to
            $SequenceSimilarity$ at index $Current$
        end for
        Add $Current$ to Priority Queue $Maximum$
        $Maximum$ checks the Sequence Similarity Scores; uses
        Weighted Edges for tie-breaking
    end for
    $MaxClique \leftarrow$ Null
    $CurrentClique \leftarrow$ Null
    for all "Top 10 Vertices in $Maximum$" do
        $TopVertex \leftarrow$ Head of $Maximum$
        for all "Adjacent Vertices of $TopVertex$" do
            $TempVertex \leftarrow$ Current adjacent vertex
            for all "Adjacent Vertices of $TempVertex$" do
                Count number of Vertices that are connected to
                $TopVertex$
                Total count will serve as the $ConnectiontoNeighborhood$
                value of $TempVertex$
            end for
            Add $TempVertex$ to Priority Queue $Neighbors$
            $Neighbors$ checks the Sequence Similarity Scores; uses the
            sum of Weighted Edges and Connection to Neighborhood
            for tie-breaking
        end for
        Add $TopVertex$ to $CurrentClique$
        for all "Vertices in $Neighbors$" do
            $Vertex \leftarrow$ Current vertex
            if "$Vertex$ is adjacent to all vertices in $CurrentClique$"
            then
                Add $Vertex$ to $CurrentClique$
            end if
        end for
        if "Total Similarity Score $CurrentClique$ is greater than or
        equal to that of $MaxClique$" then
            $MaxClique \leftarrow CurrentClique$
        end if
    end for
    Output all vertices in $MaxClique$
```

generated, it is highly unlikely that a subgraph larger than G_c will be generated between the two graphs in the datasets - and thus, ensuring that the actual MCS in the dataset remains to be the original seed graph.

Finally, G_d refers to the distraction subgraph to be added to the graph. In addition to the graph characteristics, the algorithm also accepts two parameters *as and sd*, which respectively pertain to the average similarity score and the standard deviation of the similarity scores.

3.5.2 Seed Graphs (G_c). The seed graphs are generated using an algorithm that accepts the parameter *n* referring to the number of vertices, and *m* referring to the multiplier for the number of edges based on *n*. Similarly to the main graph generation algorithm, the edges are generated randomly, thus differing for each test case.

3.5.3 Distraction Graphs G_d. The distraction graphs are formulated as attempts to direct the MCS away from detecting G_c as the largest common subgraph in each dataset. The distraction graphs are manually created within the context of MCS algorithm's logic, and primarily handles the edge cases that are less likely to be present by datasets created by the graph generation algorithm.

3.5.4 Sequence Similarity. Each dataset created by the graph generation algorithm includes a sequence similarity file, replicating the format in the [17] datasets. The sequence similarity files are present in order for the MCS algorithm to break ties, as well as direct the algorithm into providing more accurate results. The sequence similarity indexes are defined by the *as* and *sd* parameters provided in the graph generation algorithm. Vertices belonging to G_c are mapped the highest similarity indexes. Vertices with degrees within 1 of each other are mapped with moderate similarity indexes, while vertices with degrees of 0 are mapped with low similarity indexes. Other vertex mappings are not given similarity indexes, and thus not recognized as similar in the MCS algorithm.

3.6 Dataset Classification

Four dataset variants were created and generated, each with different parameters to test the flexibility of the MCS algorithm. Each dataset comprises of 24 test cases with their corresponding sequence similarity files. Each of these datasets and the seed graphs used were generated using the methods described in this section. In all datasets, the average similarity score is 200, while the standard deviation of the similarity scores is 50.

3.6.1 Dataset A: Baseline. This dataset serves as the baseline, generating two graphs that are structurally different yet share the same characteristics (eg. Number of vertices and edges).

Table 2: Baseline

Attribute	Amount
Vertices in Seed Graph	50
Vertices in G1 and G2	100
Crossing Edges	80
Inter Edges	60
Distraction Graphs	N/A

3.6.2 Dataset B: Baseline with Distractions. This dataset takes after the base dataset, with the addition of distraction graphs in order to try directing the algorithm away from retrieving the correct answer. G1 and G2 are exactly the same as those found in Dataset A, with the addition of distraction graphs.

Table 3: Baseline with Distractions

Attribute	Amount
Vertices in Seed Graph	50
Vertices in G1 and G2	100
Crossing Edges	80
Inter Edges	60
Distraction Graphs	1 on each graph

3.6.3 Dataset C: Contrasting Structures. This dataset consists of two structurally and characteristically different graphs in order to direct the graph towards finding the seed graph without getting misdirected by the addition of new vertices.

Table 4: Contrasting Structures

Attribute	Amount
Vertices in Seed Graph	50
Vertices in G1	100
Vertices in G2	70
Crossing Edges in G1	50
Crossing Edges in G2	100
Inter Edges in G1	50
Inter Edges in G2	100
Distraction Graphs	N/A

3.6.4 Dataset D: Contrasting Structures with Distractions. This dataset takes after the dataset C, with the addition of distraction graphs in order to try directing the algorithm away from retrieving the correct answer. Similarly to Datasets A and B, G1 and G2 in this dataset are exactly the same as those found in Dataset C, with the addition of distraction graphs.

Table 5: Contrasting Structures with Distractions

Attribute	Amount
Vertices in Seed Graph	50
Vertices in G1	100
Vertices in G2	70
Crossing Edges in G1	50
Crossing Edges in G2	100
Inter Edges in G1	50
Inter Edges in G2	100
Distraction Graphs	1 on each graph

3.7 Evaluating the Model

The results yielded by the model are quantitative in nature, and therefore measurable for evaluation and research purposes. The variables in conducting the testing using the model derived in this study are elaborated on below.

We ran all three algorithm variants on the four datasets that were created, and recorded their results using certain measurements of accuracy. The primary metrics used to analyze the results of the mappings provided by the algorithms are as follows:

3.7.1 Mean percentage mapping of results against the seed graph. The mean percentage mapping against the seed graph takes the total number of vertices of an output clique and then divides it by the number of common vertices in the common subgraph. This metric focuses on the number of the output and how it matches up against the seed graph. This is a measure of the quantity of the output.

$$Score = Output_n / GC_n * 100$$

3.7.2 Mean percentage mapping of results against the entire graph. The second metric is identical to the first one but instead of being compared to the size of the seed graph, the number of vertices of an output clique is divided by the number of vertices in the original graph. Like the first metric, this is a measure of quantity.

$$Score = Output_n / G_n * 100$$

3.7.3 Mean percentage mapping of exact matches. The third metric is focused more on the accuracy of the output. In the output clique, it checks the number of matches that exactly correspond to the matches in the seed graph. The seed graph serves as an answer key that the outputs strive to match. This metric, then, is not much for quantity but for quality of results.

4 RESULTS AND DISCUSSION

4.1 Results

The following tables show the results gathered from running the different algorithm variants on the four specified datasets.

Table 6: Mappings for Dataset A (Baseline)

	Against Seed Graph		Against Entire Graph		Seed Graph Matches	
	Percent Mapped	Standard Dev	Percent Mapped	Standard Dev	Percent Matched	Standard Dev
Variant 1	91.17	2.50	45.58	2.50	1.33	1.62
Variant 2	91.17	2.50	45.58	2.50	3.75	2.66
Variant 3	103.58	7.15	51.79	3.58	68.00	11.06

Table 7: Mappings for Dataset B (Baseline with Distractions)

	Against Seed Graph		Against Entire Graph		Seed Graph Matches	
	Percent Mapped	Standard Dev	Percent Mapped	Standard Dev	Percent Matched	Standard Dev
Variant 1	132.33	8.86	66.17	4.43	1.00	1.32
Variant 2	132.33	8.86	66.17	4.43	2.08	1.38
Variant 3	145	10.01	72.5	5.00	66.92	9.67

Table 8: Mappings for Dataset C (Contrasting Structures)

	Against Seed Graph		Against Entire Graph		Seed Graph Matches	
	Percent Mapped	Standard Dev	Percent Mapped	Standard Dev	Percent Matched	Standard Dev
Variant 1	46.67	2.26	28.33	1.13	0.08	0.41
Variant 2	44.75	3.33	27.17	2.02	0.5	1.06
Variant 3	98.75	6.35	59.96	3.85	94.92	8.21

Table 9: Mappings for Dataset D (Contrasting Structures with Distractions)

	Against Seed Graph		Against Entire Graph		Seed Graph Matches	
	Percent Mapped	Standard Dev	Percent Mapped	Standard Dev	Percent Matched	Standard Dev
Variant 1	97.92	14.21	59.45	8.63	0.50	0.88
Variant 2	94.58	14.36	57.43	8.72	2.83	2.57
Variant 3	150.58	14.52	91.43	8.82	95.5	7.06

4.2 Analysis

From the results, it can be seen that all three algorithms generally performed well in the quantitative metrics, as they were able to map at least 90% of the expected sizes of the seed graphs in each data set, except for dataset C. In this context, large values in the percentages of seed graphs mapped implies good performance in the algorithm used as it allows for more candidate nodes to be used in matching the seed graphs in each dataset. The qualitative evaluation of the algorithm mappings depends on the actual percentages of seed graph matches, where higher percentages matched implies higher accuracies in the final results, and thus providing the benchmark for the performances of the algorithms. Thus, given the values in the seed graph matches, it can be concluded that the third variant of the algorithm works significantly better than the first two variants.

In the percentage mapping against the seed graph, there are results exceeding 100%, meaning the sizes of the output cliques are bigger than the given seed graphs. This is because the seed graph is just the initial answer set by the graph generation algorithm. The other nodes added to the seed graph are added randomly, thus giving a possibility a common subgraph larger than the original seed graph may have been created in the datasets. Thus, percentages exceeding 100% come from the additional nodes that have been added to the seed graph and considered similar in the algorithm's computations. This behaviour is expected in all algorithms, and is exhibited in their results the most in datasets B.

The results also yielded output cliques with sizes that are good for possible most common subgraphs. Besides the performance of Variant 1 and 2 on Dataset C, all other results against the entire graph had mapping values higher than 45%. The output cliques also achieved relatively good sizes at around half the sizes of the original graphs.

The first two algorithms perform noticeably worse on Dataset C. The main difference of Dataset C from the other datasets is that its two initial graphs have unequal sizes and that it does not have a distraction graph. One possible reason for the worse performance is the fact that since there are less vertices in this dataset. Even though the size of the seed graph is the same, there are less vertices in one of the two graphs. This leads to having less vertices to choose from when selecting a matching clique and is a possibility to why the size of the output would be less.

For accuracy, only the third algorithm variant performed well and by a large margin. This has to do with the fact that the third algorithm variant prioritized sequence similarity scores over weighted edges, thus yielding more exact matches. The matches in the seed graph have higher similarity scores than other matches that are not part of the seed graph.

The addition of distraction graphs in Dataset B and Dataset D did not yield very different seed graph percentages matched for the third variant of the algorithm, as the results still showed to be accurate and relatively similar (Difference ¡ 1.5 between the two batches of datasets for the seed graph matches). The algorithm did not get distracted by the distraction graphs added and yielded results with high percentages for the seed graph. This is likely due to the direction of the sequence similarity values of the nodes in the seed graph, which consistently contained values higher than those generated outside of it.

The first two variants of the algorithms, however, saw a general increase in the percentages mapped and matched in all datasets, as well as the consequent increases in the values of the standard deviations. For the first two variants, the percentages of the seed graph matched between datasets A and B decreased, which depicts a likelihood that the algorithm was misled with the addition of the distraction graphs. The opposite behaviour was exhibited between datasets C and D, which is likely due to the contrasting structures of the datasets playing a factor in the lessened likelihood of the distraction graph being picked up or validated by the algorithms.

In general, however, the addition of distraction graph has led to substantial increases in the mappings against the seed graphs and entire graphs for all variants of the algorithms. This implies that while the additional graphs added did not necessarily distract the algorithm from the seed graph, it still mapped the distraction graph to existing nodes according to their topological similarities - and thus, leading to an increase in the percentages mapped.

The standard deviations in each round of tests exhibited certain patterns; for instance, the first and second variants of the algorithm exhibited similar deviations (Difference ¡= 1.00) for all subtests of each dataset except for dataset D's seed graph matches (Difference ¡ 2), exhibiting a level of consistency in the results of both variants. The third variant generally has higher standard deviations than the former algorithms (Except for mappings in dataset D against seed graphs, in which the difference ¡ 0.3), which is likely due to the large differences in the algorithmic structure, as exhibited in the improvements of its general performance. As the results of the third variant exhibited much higher percentage matches and graph mappings compared to the former variants, it consequently leaves more room for deviations in its results, leading to the higher standard deviations in its results.

In general, however, the standard deviations of each algorithm increased relative to the percentages mapped and matched, exhibiting that the larger the mappings, the more variations occurred in terms of its percentages mapped and matched.

Based on these results, it can be seen that the third variant of the algorithm consisting of the full implementation of sequence similarity works best when it comes to getting accurate results compared to the other variants of the algorithm. This is because it puts more bearing on sequence similarity scores instead of weighted edges. In biological terms, the biological information of the proteins are given more importance instead of the overall topological makeup of the PPI networks.

ACKNOWLEDGMENTS

We would like to thank our panelists, Dr. Jon Fernandez, Dr. Neil Tan-Gana and Dr. Andrei Coronel, for extending their expertise and feedback to improve and tailor the scope of our study. We would also like to extend our gratitude to those who have directly or indirectly supported us in this venture.

REFERENCES

[1] Ahmet Aladag and Cesim Erten. 2013. SPINAL: Scalable Protein Interaction Network Alignment. *Bioinformatics* 29, 7 (2013), 917–924. http://bioinformatics.oxfordjournals.org/content/29/7/917.full

[2] Nir Atias and Roded Sharan. 2012. Comparative Analysis of Protein Networks: Hard Problems, Practical Solutions. *Commun. ACM* 55, 5 (May 2012), 88–97. DOI: http://dx.doi.org/10.1145/2160718.2160738

[3] V.K. Balakrishnan. 1997. *Schaum's Outlines: Graph Theory*. McGraw-Hill, New York, NY. 4 pages.

[4] Patrick Prosser Ciaran Mccreesh, Samba Ndojh Ndiaye and Christine Solnon. 2016. Clique and Con- straint Models for Maximum Common (Connected) Subgraph Problems. *HAL* (2016). https://hal.archives-ouvertes.fr/hal-01331298

[5] Vazirani Dasgupta and others. 2006. *Algorithms*. McGraw-Hill Education. 247–282 pages.

[6] Mario Vento Donatello Conte, Pasquale Foggia. 2007. Challenging Complexity of Maximum Common Subgraph Detection Algorithms: A Performance Analysis of Three Algorithms on a Wide Database of Graphs. *Journal of Graph Algorithms and Applications* 11 (2007), 99–143. http://jgaa.info/accepted/2007/ConteFoggiaVento2007.11.1.pdf

[7] Ahed Elmsallati, Connor Clark, and Jugal Kalita. 2015. Computational and Structural Biotechnology Journal. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM* (2015).

[8] Nagiza F. Samatova Faisal N. Abu-Khzam. 2007. The Maximum Common Subgraph Problem: Faster Solutions via Vertex Cover. (2007). https://www.researchgate.net/publication/4252887_The_Maximum_Common_Subgraph_Problem_Faster_Solutions_via_Vertex_Cover

[9] C. Guidobaldi C. Sansone M. Vento H. Bunke, P. Foggia. 2002. A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs. *Lecture Notes in Computer Science* 2396 (2002), 123–132.

[10] H. Matsuda and others. 1999. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science* 210, 2 (1999), 305–325. http://www.sciencedirect.com/science/article/pii/S0304397598000917

[11] M. Mongiovì and R. Sharan. 2013. Global alignment of protein-protein interaction networks. *Methods in Molecular Biology* 210 (2013), 21–34.

[12] V. Memisevic O. Kuchaiev, T. Milenkovic and others. 2010. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society* 7, 50 (2010), 1341–1354. http://www.ncbi.nlm.nih.gov/pubmed/20236959

[13] Daniel Park, Rohit Singh, Michael Baym, Chung-Shou Liao, and Bonnie Berger. 2011. IsoBase: a database of functionally related proteins across PPI networks. *Nucleic Acids Research* 39 (2011), D295–D300. http://nar.oxfordjournals.org/content/39/suppl_1/D295.long

[14] Hang Phan and Michael Sternberg. 2012. PINALOG: a novel approach to align protein interaction networks—implications for complex detection and function prediction. *Bioinformatics* 28, 9 (2012), 1239–1245. http://bioinformatics.oxfordjournals.org/content/28/9/1239.full

[15] K. Ranganathan. 1999. *A Textbook of Graph Theory*. Springer, New York, NY. 8 pages.

[16] Javier De Las Rivas and Celia Fontanillo. 2010. Protein–Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks. *PLoS Computational Biology* 6, 6 (2010).

[17] Sayed Mohammad Ebrahim Sahraeian and Byung-Jun Yoon. 2012. A Network Synthesis Model for Generating Protein Interaction Network Families. *PLoS ONE* 7, 8 (08 2012), 1–14. DOI:http://dx.doi.org/10.1371/journal.pone.0041474

[18] Kyle See and others. 2015. Exploration of Protein-Protein Interaction Mappings. (2015).

[19] Tuba Sevimoglu and Kazim Yalcin Arga. 2014. Computational and Structural Biotechnology Journal. *Cell* 11, 18 (2014), 22–27.

[20] Paul Shannon and others. 2003. Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research* 13, 11 (2003), 2498–2504.

[21] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology. In *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology (RECOMB'07)*. Springer-Verlag, Berlin, Heidelberg, 16–31. http://dl.acm.org/citation.cfm?id=1758222.1758224

[22] Ulrike Stelzl and others. 2005. A Human Protein-Protein Interaction Network: A Resource for Annotating the Proteome. *Cell* 122, 6 (2005), 957–968.

[23] Damian Szklarczyk, Andrea Franceschini, Stefan Wyder, Kristoffer Forslund, Davide Heller, Jaime Huerte-Cepas, Milan Simonovic, Alexander Roth, Alberto Santos, Kalliopi P. Tsafou, Michael Kuhn, Peer Bork, Lars J. Jensen, and Christian von Mering. 2014. STRING v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic Acids Research* (2014). DOI:http://dx.doi.org/10.1093/nar/gku1003 arXiv:http://nar.oxfordjournals.org/content/early/2014/10/28/nar.gku1003.full.pdf+html

[24] J. Zola. 2014. Constructing Similarity Graphs from Large-Scale Biological Sequence Collections. In *Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. 500–507. DOI:http://dx.doi.org/10.1109/IPDPSW.2014.63