EFFICIENT COMPUTATIONS OF INTERESTING PATHS

MARC VICUNA

A thesis submitted to the Faculty of Graduate and Post Doctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Computer Science with Collaborative Specialization in Data Science

Carleton University
Ottawa, Ontario, Canada

ABSTRACT

Given a high dimensional dataset and a function of interest defined on all points, the Mapper algorithm outputs a topologically accurate summary of any numerical dataset. This summary helps identify subpopulations with interesting properties. These subpopulations typically appear as cycles, disconnected components and flares (i.e., branching paths). The discovery of these subpopulations is unique to the Mapper algorithm. This motivates its use for data mining and dataset identification.

The interestingness score of a path is defined as a sum of its edge weights multiplied by a nonlinear function of the edge ranks. Continuing the work on interesting paths by Kalyanaraman, Kamruzzaman and Krishnamoorthy, we consider the graph form of the output of the Mapper algorithm and study three optimization problems to maximize the total interestingness score of the flares: the Max-IP problem, the k-IP problem, and the IP problem. The solution to these problems leads to automatic detection of subpopulations of interest, which greatly facilitates the use of the Mapper algorithm to practitioners.

The Max-IP problem is solved in directed acyclic graphs, but we extend the solution to a special class of graphs which is common for the Mapper algorithm.

For the k-IP problem, where the number of edges in each path is fixed to k, we show the NP-completeness proof given by Kalyanaraman, Kamruzzaman and Krishnamoorthy has gaps. We give a new NP-completeness proof of the k-IP problem for $k \geq 4$. We design three approximation algorithms, where the best approximation bound is $\frac{3}{k+1+\epsilon}$, where $\epsilon > 0$. We also design three exact algorithms with varying assumptions, namely: no assumptions, limited graph diameter and constant output size.

For the IP problem, where the number of edges in each path is not fixed, we give a proof of the NP-completeness of the IP problem. We design exact algorithms and 9/20-approximate algorithms, using various heuristics.

Topology is the science of fundamental pattern and structural relationships of event constellations.
— R. Buckminster Fuller

ACKNOWLEDGMENTS

Thanks to my supervisors Anil Maheshwari, Michiel Smid and Jean-Lou De Carufel and my friend Saeed Odak for helping me throughout the entire process, for their patience, support, insights, constructive discussions, reviews, and improvements to my ideas, proofs, heuristics, and drafts. Thanks to the examination committee for their helpful comments. Thanks to Bodhayan Roy from IIT Kharagpur for providing the initial NP-Hardness reduction for the DAEDk problem for $k > 3$. Thanks to my fiancée Speline, my family and my friends for their love and support throughout this work. Thanks to Sander Verdonschot for the LaTeX thesis template.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

TDA Topological Data Analysis

LPath Longest path measure, defined in Section 1.8.3

1 INTRODUCTION

Data has been continuously growing since 2010, preceded by the rise of the Internet of Things (IoT). Huge amounts of data are generated every day, largely collected from various types of sensors on a growing collection of connected devices [74]. As the majority of available data is unlabelled, research in unsupervised learning and data mining is highly valued for the classification, regression, clustering and summarization of unlabelled datasets.

1.1 BIG DATA ANALYTICS (BDA)

Since many terms in the domain of Big Data Analytics have been defined inconsistently throughout the literature, it is crucial to define these terms to establish the challenges and applications of the algorithms explained. The following definitions are taken from a survey [25] of the use of these terms.

- **Big data analytics** is the useful search for information in datasets of any kind that is hardly feasible using traditional methodologies, techniques, and tools due to the characteristics of the data to be analyzed and that are known as "V", which are volume, velocity, variety, veracity, variability, value and visualization, to obtain results that allow to analyze data, take current trends, optimize the use of resources and/or predict the future, in a fast and efficient manner [25].
- **Data mining** is the process of extracting data, analyzing it from many dimensions or perspectives, then producing a summary of the information in a useful form that identifies relationships within the data. There are two types of data mining: descriptive, which provides information about existing data; predictive, which makes forecasts based on the data [27].
- **Data analysis** is the collection of methods for drawing inferences from data, thus extracting global information that is generally the property of the analyzed data. [27]

Thus, big data analytics includes the search and solutions to the problems and opportunities created by big data. The algorithmic methods of big data analytics can broadly be divided in two parts: data analysis and data mining. Data analysis is closer to classical statistics, in that many fixed characteristics can be defined from a given dataset, such as an average, or an expected behaviour from a given input.

The methods of data analysis use (implicitly or explicitly) hypothesis testing to draw conclusions and insights. Data mining is the opposite of data analysis in that there is no hypothesis testing, rather data mining methods aim to automatically give and forecast information as part of Knowledge Discovery in Databases (KDD), which is the automatic identification process of valid, novel and potentially usable of pattern understanding in large databases [22].

Data mining is nowadays largely associated with machine learning, clustering and dimensionality reduction. Out of all the approaches of data mining, machine learning in Big Data Analytics has especially grown, now being applied in the healthcare industry, in anomaly detection, in cybersecurity, in data privacy & IoT, in the automobile and transportation industry, in E-commerce and many more fields [52]. However, these approaches generally approach cluster analysis geometrically, with an emphasis on size and position rather than structure or shape. Even with normalization schemes, machine learning and clustering are both sensitive to scaling. Moreover, in machine learning or clustering, a cluster is usually assumed to be a closed n-dimensional ball, meaning the region containing the cluster has no holes. In other words, the basic interesting unit of these methods is assumed unimodal (for example, multivariate normally distributed), a limitation that may heavily restrict cluster characterization. As geometrical approaches of this type grow in popularity and extensiveness, this minor problem in its early days becomes increasingly major and worth analyzing. This motivates the development of Topological Data Analysis (TDA) [6].

1.2 TOPOLOGICAL DATA ANALYSIS

In mathematics, topology is concerned with the properties of a geometric object that are preserved under continuous deformations. Often seen in opposition to geometry, where measures such as angles and distances are crucial, topology is concerned with properties of measure-independent shapes. In the two last decades, the field of computational topology has grown in applicability in many fields, such as graphics [56], robotics [57], structural biology [63] and chemistry [46]. One major subfield of interest in computational topology is the application of topological techniques to data science, that is, TDA. Algorithms of this field follow the following scheme:

1. An unlabelled multidimensional dataset is given.
2. A transformation is made to that set of points to make it a simplicial complex, a set composed of points, line segments, triangles, and other n-dimensional simplexes.

3. An insight is computed concerning the shape of the dataset, in the form of numerical outputs (data analysis) and discrete structures (data mining).

These insights relate to topological properties of the dataset, that is, to its homology. TDA is theoretically based on axioms of topology (needed for manifold analysis) and axioms of probability (needed for statistical results over data) [63]. The robust theoretical grounds of TDA make it appropriate for analysis of high dimensional and complex data. Indeed, due to the complexity in visualizing high dimensional manifolds, very few data mining techniques can characterize high dimensional shapes precisely. TDA can be considered a generalization of clustering, allowing for non-metric spaces, and thus, non-metric measures of distance, another rare capability in data mining. However, this generality has a cost: since the output of the TDA algorithm relates to abstract properties of significant clusters, interpretation of the output may be difficult in practice, and some computations often have a higher computational complexity than competing non-topological algorithms. Thus, in most cases, TDA is best suited for data exploration of difficult datasets. In the majority of research articles, using TDA, TDA is used in combination with other more classical tools in data analysis and data mining [63].

1.3 HOMOLOGY

Homology is a general way of associating a sequence of algebraic objects with other mathematical objects, such as topological spaces[1]. Homology analysis implies the classification of manifolds according to their cycles–closed loops (or more generally submanifolds) that can be drawn on a given N-dimensional manifold but not continuously deformed into each other. A homology class is thus represented by a cycle which is not the boundary of any submanifold: the cycle represents a hole, namely a hypothetical manifold whose boundary would be that cycle, but which is "not there". For example,

1. The disconnectedness of two components is a 1-dimensional hole.
2. A cycle with no interior is a 2-dimensional hole.
3. The hollowness of a sphere is a 3-dimensional hole.

The algorithmic characterization of datasets is typically done through Betti numbers, that is, the number of holes of each dimension in a given manifold [26].

1 In the case of the Mapper algorithm, the manifolds are all simplicial complexes.

1.4 PERSISTENT HOMOLOGY

The most common problem in TDA is to assess the persistent homology of datasets. This generalization of homology can be originally attributed to Edelsbrunner, Letscher and Zomorodian [19, 20].

Homology can practically be seen as the characterization of manifolds. In persistent homology, discrete manifolds are usually expressed as simplicial complexes. A simplex is the simplest polytope of some dimension:

1. a 0-dimensional simplex is a point,
2. a 1-dimensional simplex is a line segment,
3. a 2-dimensional simplex is a triangle,
4. a 3-dimensional simplex is a tetrahedron.

A simplicial complex is a set of simplexes.

The idea of persistent homology is best expressed by the following algorithm. Consider a set D of points of N dimensions. For each point, a N-dimensional ball is created, of radius r, where r goes from 0 to ∞. Whenever i N-balls intersect, an $(i-1)$-dimensional simplex is formed using the centres of these balls. As a result, each point starts as a 0-dimensional simplex. At the end, all points form one single simplex.

Through this process, many manifolds are created. For example, in two dimensions, the disks of radius r in Figure 1.1 lead to the simplicial complex of Figure 1.2. In three dimensions, the balls of radius r in Figure 1.3 lead to the simplicial complex of Figure 1.4. This process preserves the homology of a dataset relative to a given r. However, this does not intuitively lead to a notion of homology of the entire dataset.

This is where the notion of persistence is necessary. By varying r, some features of a dataset may disappear or evolve into different types of holes. For example, for two points becoming a line, one may consider the first point to have become a line, while the other point is considered to have "disappeared". Thus, the holes of the simplicial complex that are maintained over significant intervals of r are considered persistent.

Persistent homology algorithms are the main contenders to the Mapper algorithm, seen in the next chapter. Together, persistent homology and Mapper are the two most popular algorithms of TDA.

1.5 HIGH CLUSTER ANALYSIS

Unsupervised clustering and TDA share many common points. In fact, discrete homology may be defined as a generalization of clustering. Unsupervised clustering algorithms aim to group a set of N-dimensional

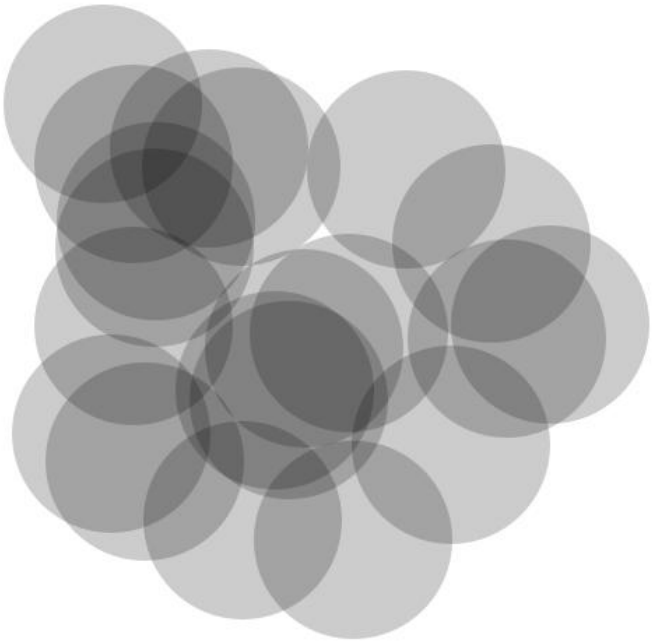


Figure 1.1: Growth of N-balls in 2 dimensions

points in multiple subsets (clusters) where the elements are similar according to some function, often called distance in the context of metric spaces.

Given a dataset, the analysis of 0-dimensional holes of a simplicial complex formed at r is equivalent to the notion of clustering, where r is a bound on the allowed dissimilarity between points within a cluster. As such, the analysis of holes of greater dimensional degree reveals more information about the clusters formed. Note, the association between clusters and their corresponding holes requires more information than Betti numbers.

1.6 REEB GRAPHS

A Reeb graph is a graph representing the progression in level sets of a real-valued function f on a continuous manifold [61]. Originally, this concept was introduced as a tool in Morse Theory [55], as the discrete changes in the homology of continuous level sets may be inferred from the analysis of critical points in f.

For data analysis purposes, Reeb graphs may be useful for the homology analysis of datasets. Indeed, the analysis of the Reeb graph for a discrete dataset is useful [14], and the complexity used to construct Reeb graphs is relatively low [18]. However, the complexity of large datasets makes the interpretability of Reeb graphs difficult and sensitive to outliers. Thus, to solve these issues, the following section explains the Mapper algorithm.

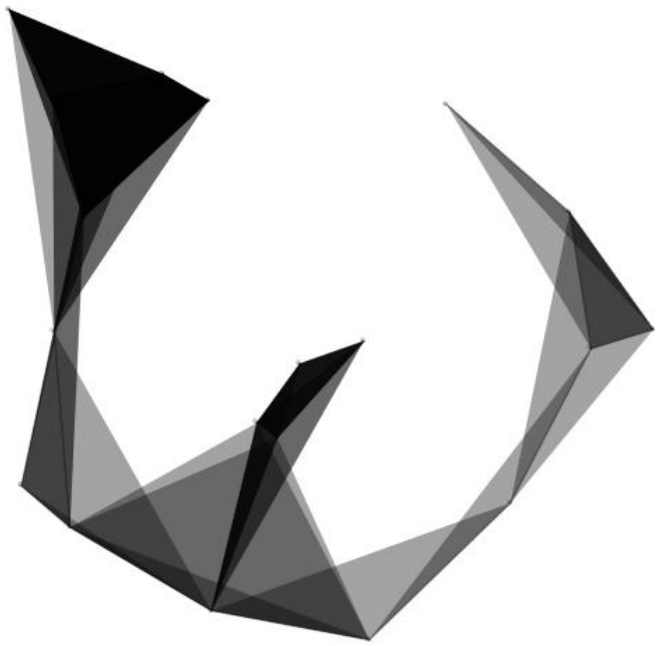


Figure 1.2: The simplicial complex in 2 dimensions of Figure 1.1

1.7 MAPPER

The Mapper algorithm was invented in 2007 by Singh, Mémoli and Carlsson in *Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition* [62].

From the context of TDA, this section covers the motivation of this algorithm, the algorithm itself and important applications.

1.7.1 *Motivation*

Like persistent homology, the goal of the Mapper algorithm is the qualitative analysis, simplification, and visualization of high dimensional data sets over some function of interest. The Mapper algorithm expresses an empirical approach to topological data analysis, solving the practical concerns of previous algorithms. As such, it is useful to note characteristics of competing algorithms that contrast with Mapper.

1. Classical clustering fails to capture topological features of clusters.
2. Reeb graphs fail to summarize datasets intuitively.
3. Persistent Homology and discrete Reeb graph algorithms are computationally expensive for large datasets.

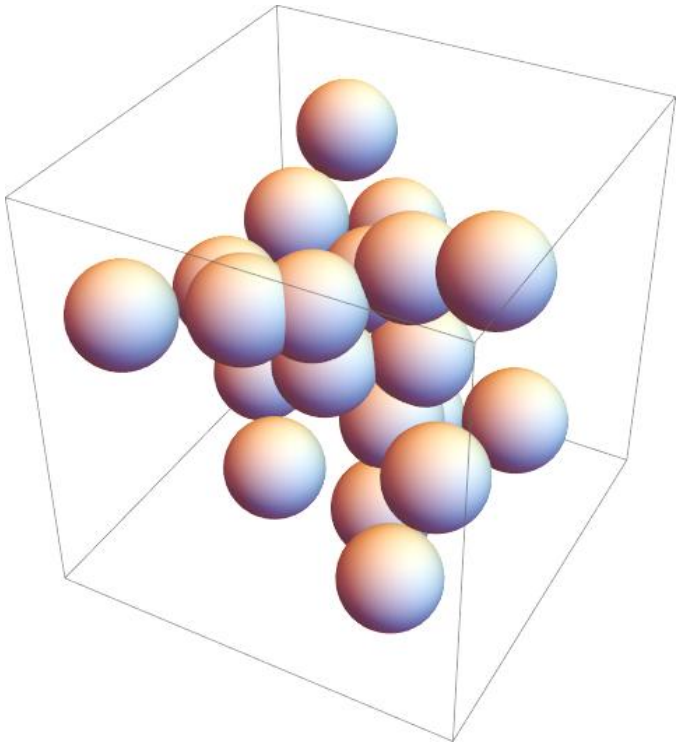


Figure 1.3: Growth of N-balls in 3 dimensions

4. Persistent Homology has a numerical output which makes any topological hole detected difficult to associate to any cluster or position.

The Mapper algorithm solves those issues simultaneously by combining persistent homology, clustering and Reeb graphs.

1.7.2 *Algorithm*

Consider a multidimensional dataset X and a function w which maps each point $x \in X$ to some value $w(x) \in \mathbb{R}$. Consider as parameters to the algorithm a clustering algorithm, an interval length $l \in \mathbb{R}$ and an overlap $p < l$. This can be summarized as follows: the topological space U in which X is embedded is divided into multiple subspaces of equal width l based on w. Each subspace shares with each neighbouring subspace a smaller subspace of width p. Then, the Mapper algorithm outputs a simplicial complex summarizing the topology and persistent homology of X within U based on w.

The complexity of the Mapper algorithm is dependent on the clustering method used. If the clustering method used is of $c(n)$ time complexity, where n is the size of the dataset X, then the Mapper algorithm is of $O(\frac{b-a}{s-l}c(n))$, which is in the worst case $O(n \cdot c(n))$. In the case where each cluster has only one point, the Mapper algorithm is equivalent to calculating the discrete Reeb graph of X. This implies

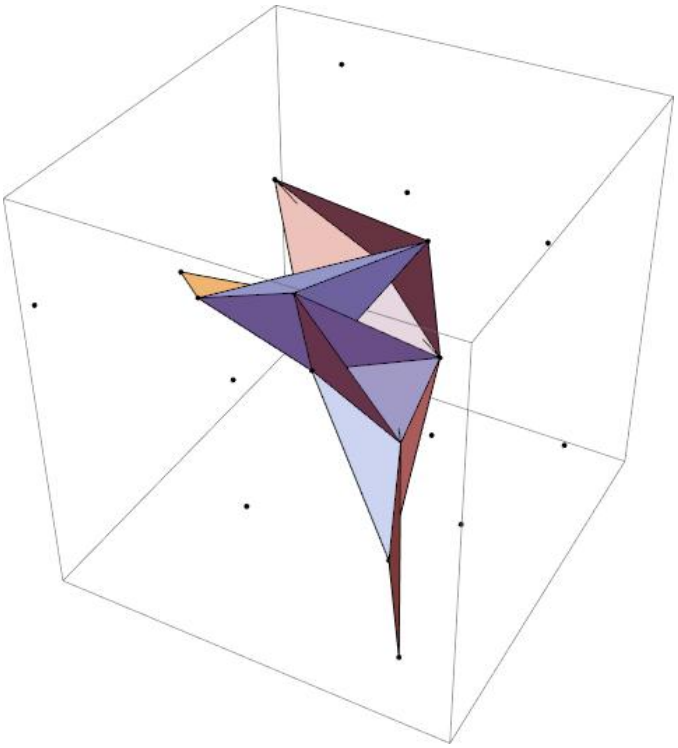


Figure 1.4: The simplicial complex in 3 dimensions of Figure 1.3

the Mapper algorithm preserves the homology of X in those cases. However, clusters of Mapper are expected to be much larger than a single point, maximizing the differentiation of clusters over some metric of similarity. Thus, while the exact homology of Reeb graphs may not be maintained in the Mapper output, the persistent homology of the Reeb graph is maintained in the Mapper output, that is, the topology of significantly different topological features are maintained in the Mapper output through consistent clustering over level sets. Moreover, unlike Reeb graphs, the persistent homology is statistically robust, because of the underlying clustering algorithm being statistically robust. In fact, depending on the application, the qualities of the Mapper algorithm directly relate to the qualities of the clustering algorithm chosen and the resolution chosen, which encapsulates both the length of intervals l and their overlap p.

1.7.3 *Applications*

As the most used algorithmic tool in TDA, the Mapper algorithm has a large array of applications. We note here the most common fields of application: neuroscience [36, 50, 65], geography [66], organic chemistry [7], synthetic aperture radar (SAR) data [33], engineering [29], finance [15], 3D image processing [9], social networks [16], genetics [38], phenomics [77], visualization [75] and neural networks [5].

Algorithm 1 Mapper Algorithm

Require: Dataset X, Function $w : X \to \mathbb{R}$, Interval length l, Overlap $p < l$

Measure the minimum value a and maximum value b of $w(x)$ for all $x \in X$
$G \leftarrow \{\}$
$s \leftarrow a + l$
while $s - l \leq b$ **do**
    Cluster all points of $x_j \in X$ such that $s - l \leq w(x_j) \leq s$
    for Each cluster c formed, ordered by $w(x)$ **do**
        Add an edge from a cluster in G to c when some point in c is shared between any two clusters.
        Add c as a vertex in G.
    end for
    $s \leftarrow s + l - p$
end while
return G

In those applications, the Mapper algorithm is used with large datasets of high dimensionality where more conventional data mining techniques do not provide a more profound understanding of the data. Applications typically go through three steps specific to the Mapper algorithm:

1. Design domain-appropriate functions of interest, also called filters, for the Mapper algorithm.

2. Run the Mapper algorithm on a varied set of parameters based on the domain-specific notion of significance.

3. Analyze through trial-and-error submanifolds of the output of the Mapper algorithm, for subpopulations with monotonic variation, and identify the set of attributes correlated to the subpopulations.

   The Mapper output is used for summarization, identification, prediction, and categorization. Unlike many modern data mining algorithms, the Mapper algorithm does not lead to pattern recognition over the entire dataset, but allows for pattern recognition over isolated parts of the data.

1.8 OUTPUT CHARACTERIZATION

The Mapper algorithm outputs a simplicial complex that is called the Mapper, summarizing the topological features of the dataset. In some versions of the Mapper algorithm, the Mapper may be post-processed using simplex contraction (similar to edge contraction), since that operation may simplify the Mapper without modifying the

homology of the Mapper. To analyze the Mapper, a weighted directed graph $G = (V, E)$ representation of the 1-skeleton of the Mapper is constructed, with additional information in the form of weights. This is described in the Section 2.1.1 of [41]. Note, the 1-skeleton is the graph having the 0-dimensional simplexes as vertices and 1-dimensional simplexes as edges.

This section covers the characterization of the output graph of the Mapper algorithm, starting with its formulation, then all its potential simplifying assumptions.

1.8.1 *Graph Formulation*

Consider the simplicial complex Mapper, the output of the Mapper algorithm, the set of points X corresponding to it. The graph G is defined as the 1-skeleton of the Mapper, such that V is the set of points (clusters) in Mapper and E is the set of lines in Mapper. Consider a random ordering Φ of V. A tolerance τ is given for the graph G. Define $X(v_i)$ to be the set of points in X included in the cluster corresponding to $v_i \in V$. The weight of vertices is defined as the average $w(x)$ value of each $x \in X(v_i)$:

$$w(v_i) = \frac{\Sigma_{x \in X(v_i)} w(x)}{|X(v_i)|} \tag{1.1}$$

Consider the two vertices of any edge e as $v_{e,1}$ and $v_{e,2}$. Once the weight of any vertex is calculated, the weight of an edge e is defined as the absolute difference between the weights of its corresponding vertices: $w(e) = |w(v_{e,1}) - w(v_{e,2})|$. Finally, the direction of the edges in G is determined by the following rule:

1. If $|w(v_{e,1}) - w(v_{e,2})| < \tau$, then the edge is from $v_{e,1}$ to $v_{e,2}$ and another edge from $v_{e,2}$ to $v_{e,1}$ (bidirectional).
2. Otherwise, if $w(v_{e,1}) < w(v_{e,2})$, then the edge is from $v_{e,1}$ to $v_{e,2}$.
3. Otherwise, if $w(v_{e,1}) > w(v_{e,2})$, then the edge is from $v_{e,2}$ to $v_{e,1}$.
4. Otherwise, $w(v_{e,1}) = w(v_{e,2})$, and the direction of the edge follows the ordering Φ.

This is the construction studied in this present work. This graph is referred to as the output graph of the Mapper algorithm, and is the structure that is being studied and considered throughout all practical considerations and experimental results.

Example 1.8.1. *Consider the graph of five vertices, where the weights of the vertices are* $0.4, 0.7, 1.3, 1.4, 2.1$.

If $\tau = 0$, *then no double edges are formed, see Figure 1.5.*

If $\tau = 0.8$, *then there are three new edges formed, as* $1.4 - 0.7 < \tau$, $1.3 - 0.7 < \tau$ *and* $2.1 - 1.4 < \tau$, *see Figure 1.6.*

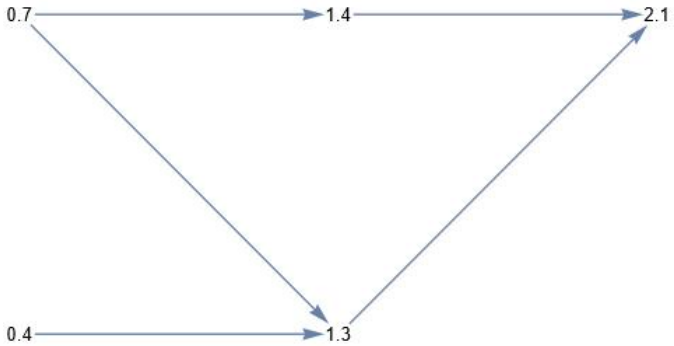


Figure 1.5: output graph example where $\tau = 0$

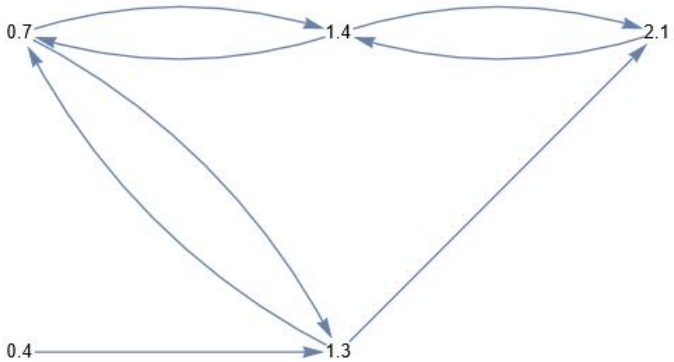


Figure 1.6: output graph example where $\tau = 0.8$

If $\tau = 1.0$*, then there are two new edges formed, as* $1.3 - 0.4 < \tau$*, and* $2.1 - 1.3 < \tau$*, see Figure 1.7.*

In practice, some assumptions may lead to better algorithms. The following subsections define the main assumptions that may be considered.

1.8.2 *Acyclic Graphs*

A directed acyclic graph (DAG) is a directed graph with no directed cycles. Note, a topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph [39]. In the general case, the output graph of the Mapper algorithm may have bidirectional edges, as mentioned in the preceding case 1, which creates cycles. The case where $\tau = 0$ is an exception: all edges $v_a \to v_b$ following the condition $w_{v_1} \leq w_{v_2}$. This implies there is a topological ordering for all vertices, which implies the graph is acyclic. Thus, G must be a directed acyclic graph (DAG).

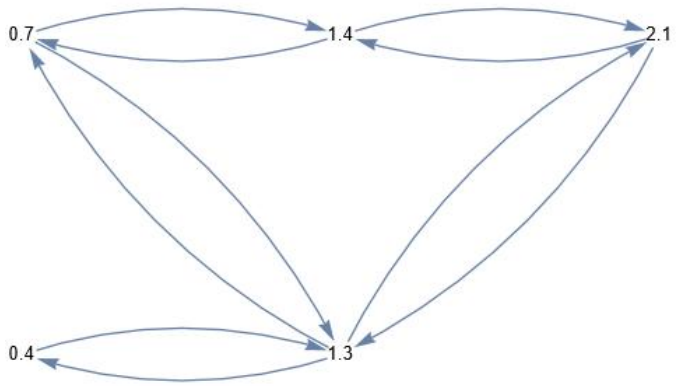


Figure 1.7: output graph example where $\tau = 1.0$

1.8.3 *Longest Path Measure (LPath)*

Given a directed acyclic graph G, the *LPath* of a vertex v_i, denoted $LPath(v_i)$, is defined as the number of edges of the longest path whose first vertex is v_i.

Lemma 1.8.2. *Consider a directed acyclic graph $G = (V, E)$ where $|V| = n$ and $|E| = m$. The LPath metric dictionary can be constructed in $O(n + m)$ time.*

Algorithm 2 Longest Path Dictionary

Require: Graph $G = (V, E)$
1: $LPath \leftarrow \{\}$
2: **for** each vertex v_i in the reverse topological order of G **do**
3: **if** outdegree(v_i)$= 0$ **then**
4: $LPath[v_i] \leftarrow 0$
5: **else**
6: $LPath[v_i] \leftarrow 1 + \max_{(v_i \to v_j) \in E}(LPath[j])$
7: **end if**
8: **end for**
9: **return** $LPath$

Proof. The reverse topological sorting of G is found in $O(n + m)$. The *LPath* dictionary algorithm considers at most $O(n + m)$ elements. Thus, the *LPath* dictionary can be constructed in $O(n + m)$ time. □

1.8.4 *Multilayered Graphs*

In the case of the output graph of the Mapper algorithm, it may be useful to consider that each cluster analyzed is found on a level set

of clusters. As such, a graph layer is defined as the set of all vertices representing clusters from the same level set.

1.8.4.1 *Definition*

A digraph $G = (V, E)$ is multilayered if and only if there exists a pairwise disjoint set of vertices $\{L_1, L_2, ..., L_\zeta\}$ such that $V = \cup_{i=1}^{\zeta} L_i$ and for each edge $v \to u$ in E, $v \in L_j$ and $u \in L_{j+1}$ or $v \in L_{j+1}$ and $u \in L_j$ for some j.

1.8.4.2 *Assumptions*

The first assumption is that the clustering algorithm chosen classifies each point as being part of only one cluster. As a consequence, within vertices of the same cluster, there are no edges, as no points in X are shared by two clusters of the same level set.

The second assumption is that $p < l/2$, that is, the length of the overlap between layers is less than half the interval length l, as defined in Section 1.7.2. This implies all edges are only between 2 consecutive layers. By contradiction, if an edge is between 2 nonconsecutive layers, then consider the domain of the clusters can be denoted $[0, l], [l - p, 2l - p]$ and $[2l - 2p, 3l - 2p]$. Some cluster must be shared between $[0, l]$ and $[2l - 2p, 3l - 2p]$. Thus,

$$l \geq 2l - 2p \tag{1.2}$$

$$p \geq l/2 \tag{1.3}$$

which is a contradiction.

Thus, using the assumption $p < l/2$, it is realistic to consider multilayered graphs.

1.8.5 *Weight-Balanced Graphs*

Assume the output graph of the Mapper algorithm is a multilayered graph. Consider two arbitrary consecutive layers L_1 and L_2 of domain $[a, a + l]$ and $[a + l - p, a + 2l - p]$ respectively. Since the weight of any vertex in L_1 must be between a and $a + l$ and the weight of any vertex in L_2 must be between $a + l - p$ and $a + 2l - p$, then the weight of an arbitrary edge e between L_1 and L_2 must be between 0 and $2l - p$. Likewise, the sum of weights of a path of k edges is between $(k - 1)(l - p)$ and $l + (l - p)k$.

A weighted multilayered digraph $G = (V, E)$ is weight-balanced if and only if the sum of weights of a path P of k edges is between $(k - 1)(l - p)$ and $l + (l - p)k$.

$$\forall P(\Sigma_{i=1}^{k} w_{e_i \in P} \in [(k - 1)(l - p), l + (l - p)k]) \tag{1.4}$$

Clearly, in the case of the multilayered output graph of the Mapper algorithm, this property is maintained.

1.8.6 *Filter functions*

The clustering function $w(x) : X \to \mathbb{R}$ of the output graph of the Mapper algorithm may be extended to any number of dimensions. This can be done through the use of filter functions, often considered attributes of interest for all points of X. As such, the filter function $f_j(x) : X \to \mathbb{R}$ value of any vertex v_i in G is defined as the average of the filter values of points included in the cluster represented by v_i:

$$f_j(v_i) = \frac{\Sigma_{x \in X(v_i)} f_j(x)}{|X(v_i)|} \tag{1.5}$$

Consider a set of h filter functions $\{f_1, f_2, \cdots, f_h\}$. From each filter value, a h-bit binary signature $Sig(e) = b_1 b_2 b_3 ... b_h$ for each edge $u \to v$, where $b_i = 1$ if $f_i(u) \leq f_j(v)$, and $b_i = 0$ otherwise. The study of path partition can be made on the subgraphs of G where all edges have the same signature.

While this may be considered by the practitioners as a complete application of the algorithms, a graph using diverse h-bit binary signatures can clearly be computed as many graphs of identical signatures and these solutions may be joined optimally. As such, since this does not constitute a concern computationally, the signatures of all graphs are assumed to be identical.

1.9 INTERESTINGNESS

The term *interestingness* in the context of the Mapper algorithm refers to subpopulations of X with topological properties. Those properties include cycles, three-dimensional holes, and most importantly for this work, *flares*. Flares may be visually defined as consecutive sets of points of monotonic variation that deviate from the main sequence of sets of points with monotonic variation. The concept of flare is entirely motivated by practice: for two-dimensional or three-dimensional sets of points, it corresponds to a frequently observable shape "manually" studied in data mining.

Let G be the output graph of the Mapper algorithm. The *interestingness score* of a k-path P in the output graph of the Mapper algorithm is defined as:

$$I(P) = \Sigma_{r=1}^{k} w(e_r) \cdot \log_2(1 + r) \tag{1.6}$$

Note, all logarithms in this work are base 2. The term "interesting path" refers to a weighted path and its corresponding interestingness score.

Example 1.9.1. *Consider the path of 3 edges, with vertex weights of 0, 1, 3 and 3.5. This implies the edge weights are of 1, 2 and 0.5, as shown in Figure 1.8. The interestingness score of this graph is:*

Figure 1.8: A weighted interesting 3-path

$$I(P) = \Sigma_{r=1}^{k} w(e_r) \cdot \log_2(1+r) \tag{1.7}$$
$$= 1\log_2 2 + 2\log_2 3 + 0.5\log_2 4 \tag{1.8}$$

This formulation of path interestingness has two major goals:

1. The weight factor ensures the paths of high interestingness scores cover a significant portion of the clustering function w.
2. The $\log_2(1+r)$ factor ensures the path is lightly biased to heavy-tail paths, as it is typical of flares that the monotonic increase in variables is most pronounced at the end of a path.

1.9.1 *Interestingness Score of Unweighted Graphs*

The study of unweighted graphs in this work is motivated by weight-balanced graphs, where $w_e \in [0, 2l-p] |\ \forall e \in E$. In the case of unweighted graphs, that is, in the case $\forall(e_i \in E)(w(e_i) = 1)$, the interestingness score formula can be simplified using the log property of the addition of log terms:

$$I(P) = \Sigma_{r=1}^{k} \log_2(1+r) = \log_2((1+k)!) \tag{1.9}$$

Theorem 1.9.2. *For weight-balanced graphs, for an arbitrary path of size k, the worst approximation between paths of the same length is $(\frac{\Sigma_{r=2-(k \mod 2)}^{\lceil k/2 \rceil} \log_2(1+r)}{\Sigma_{r=1+(k \mod 2)}^{\lceil k/2 \rceil} \log_2(1+r)})^{(-1)^k}$, where the approximation factor varies from $\frac{1}{\log_2(3)}$ to 1 as k increases.*

This expression arises from the analysis of the critical points of the equation 1.6. To maximize the difference in interestingness score of two weight-balanced paths of length k, consider the edges to be numbered from 1 to k. The theoretical limit for the maximum difference in interestingness score is found when the first path has a weight of $2-\epsilon$ for odd edges and a weight of ϵ for even edges, and the second path has a weight of $2-\epsilon$ for even edges and a weight of ϵ for odd edges, where ϵ is a small positive number. Note the exponent $(-1)^k$ expresses the alternation between each path being the largest.

Overall, this property justifies the use of the unweighted assumption for weighted weight-balanced graphs, as for $k > 4$, the approximation cost is relatively low.

1.9.2 *Optimization Problems*

The optimization problems relative to interesting paths were first defined in [41]. These constitute the main subject of this thesis. For all problems, consider the output graph G of the Mapper algorithm, with vertices V of size n and edges E of size m, where each edge has a weight $w(e_i) \geq 0$.

- **Max IP Problem**: Find the path P in G such that its interestingness score $I(P)$ is maximized.
- **k-IP Problem**: Find the edge-disjoint set P in G of k-paths (paths of length exactly k) such that the sum of their interestingness score is maximized.
- **IP Problem**: Find the edge-disjoint set P in G of paths such that the sum of their interestingness score is maximized.

1.10 CONTRIBUTIONS

Based on the framework of interestingness quantification given in [41], we propose multiple ideas to ease subpopulation discovery.

We prove the NP-complete Max-IP problem may be avoided while allowing some tolerance, which is required in practice.

We show that the preceding NP-completeness proof for the k-IP problem on directed acyclic graphs has gaps. We offer a new NP-completeness proof of the k-IP problem for $k \geq 4$ on directed acyclic graphs. We design multiple approximate and exact algorithms to solve the k-IP problem of various complexities and approximation bounds.

We show that the IP-problem is NP-complete and design multiple approximate algorithms to solve the IP problem. Preliminary results based on the implementation of approximation algorithms are given for the k-IP and IP problems.

1.11 ORGANIZATION

Chapter 2 covers important works related to the algorithms and proofs developed. The main chapters are the 3 problems mentioned above: chapter 3 explores the Max-IP problem, chapter 4 examines the k-IP problem, and chapter 5 examines the IP problem. Each of these chapters includes the definition, motivation, complexity, and algorithms for this problem. In chapter 6, we conclude with the significance and utility of our contributions.

2

RELATED WORKS

The study of interesting paths may be considered as the interaction between three active fields of research: data mining, topological data analysis and subgraph decomposition algorithms. The notion of a weight-scheme to score the value of a pattern comes from data mining, the notion of using the Mapper algorithm as an equivalent representation of a dataset comes from topological data analysis and the notion of using path decomposition to represent subpopulations of interest can be traced back to subgraph decomposition algorithms.

This section discusses the major references and relevant papers. The first two papers address the concept of interesting paths in the context of the output graph of the Mapper algorithm. The third paper addresses another measure for paths in the context of the output graph of the Mapper algorithm. The fourth and fifth articles address the problem of path partitions and provide context to theoretical proofs.

2.1 INTERESTING PATHS IN THE MAPPER

The article by Kalyanaraman, Kamruzzaman and Krishnamoorthy [41] is the main reference of this work, as it defines the output graph of the Mapper algorithm, path interestingness score and the Max-IP, k-IP and IP problems. It provides an algorithm for the Max-IP on directed acyclic graphs, proves the NP-completeness of the Max-IP on general digraphs and of the k-IP on directed acyclic graphs[1] and proposes a heuristic for an approximation of the IP problem.

As a whole, this article provides a strong base for the theoretical motivation of these problems, but the limit and unknown approximation factors of the problems and algorithms proposed make it difficult for practitioners to study path interestingness.

Our work answers multiple open questions proposed in the discussion section of [41], namely the NP-completeness of the IP problem, the NP-completeness of unweighted k-IP problem, approximation algorithms for the k-IP and IP problem, as well as the performance of the Longest Path Heuristic.

2.1.1 *Topological Data Analysis for Computational Phenomics: Algorithms and Applications*

Interesting Paths in the Mapper is one of the articles that led to the completion of the PhD thesis on computational phenomics by Kam-

1 We argue this proof has gaps.

ruzzaman [42]. In this thesis, a heuristic is given for the IP, k-IP and At-Least-k-IP problems, all on directed acyclic graphs. The usefulness of the interestingness score metric is motivated by its performance in phenomics datasets.

2.2 HYPPO-X

In terms of software, the set of tools for interaction with the Mapper output graph, such as *Mapper Interactive* [76] have been extended with the Hyppo-X software [43], which allows users to interactively consider interesting paths. This implementation mainly uses the Longest Path heuristic to solve the IP problem. This software specifically targets phenomics, and multiple researchers in phenomics have had success using this tool for data mining [40].

The Hyppo-X software limits its algorithms to only the IP heuristic. With better algorithms for the calculation of interestingness score, the Hyppo-X software may be extended beyond its current form for the Max-IP or the k-IP approximations.

2.3 FINDING MAXIMUM DISJOINT K-PATHS

The problem of finding disjoint k-paths can be traced back to different research articles from Shiloach, Itai and Perl [37, 58–60]. The article *The Complexity of Finding Maximum Disjoint Paths with Length Constraints* [37] constitutes the base for most research on length constrained disjoints paths. This paper establishes 4 problems that relate to the k-IP problem:

1. VDK, the vertex-disjoint exactly-k paths problem, where the goal is to find vertex-disjoint paths of length exactly k.
2. VBK, the vertex-disjoint bounded-k paths problem, where the goal is to find vertex-disjoint paths of length at most k.
3. EDK, the edge-disjoint exactly-k paths problem, where the goal is to find edge-disjoint paths of length exactly k.
4. EBK, the edge-disjoint bounded-k paths problem, where the goal is to find edge-disjoint paths of length at most k.

Edge-disjoint k-path partition problems may be considered of a higher complexity than vertex-disjoint k-path partition problems. Indeed, to solve any vertex-disjoint k-path partition problem on a graph G, one may use an edge to represent any vertex in G, transforming any vertex-disjoint k-path partition problem into an edge-disjoint k-path partition problem. However, transforming any edge-disjoint k-path partition problem into a vertex-disjoint k-path partition problem is not possible. Note that vertex-disjoint path partition problems may be considered a subset of edge-disjoint path partition problems [37].

The EDK problem is proven to be NP-hard even for undirected planar graphs [78]. In comparison, the VDK problem has well-known approximate algorithms [47].

The EDK problem is the problem that most closely resembles the k-IP problem. However, major differences are present in their input, namely:

1. The graphs studied in the EDK problem are all undirected, and the graphs studied in the k-IP problem are all directed.
2. The paths studied in the EDK problem have given start and end vertices, and the paths studied in the k-IP problem do not have given start and end vertices.

This suggests the k-IP problem is also NP-complete, and indeed, we prove its NP-completeness in Section 4.5

2.4 HOLYER'S CONJECTURE

Consider a graph $G = (V, E)$ and a subgraph H of G. An H-decomposition of a graph G is a partition of E into subgraphs isomorphic to H. In 1980, Holyer [32] conjectured that an H-decomposition of G is NP-complete whenever H is connected and has three edges or more. Dor and Tarsi [17] proved this conjecture in 1997.

Consider the case where H is a path graph of 3 edges. This theorem implies H-decomposition of G, that is, the 3-path-decomposition of G is NP-complete. This constitutes a proof of the NP-completeness of the k-IP problem in the case of unweighted directed graphs, simply by using the Holyer's Theorem. However, it does not imply the NP-completeness of the k-IP problem for unweighted directed acyclic graphs, as they are a subclass of unweighted directed graphs.

2.5 STABLE PATHS

Besides interesting paths, stable paths are also studied in the context of the output graph of the Mapper algorithm. A path P is defined to be ρ-stable if $\max(w(e)|e \in P) \leq \rho$. In other words, a path is ρ stable if all edge weights are less than or equal to ρ. Since edge weights are positive for the output graph of the Mapper algorithm, ρ is positive. In many applications using the Mapper algorithm, $\rho < p$, where p is the overlap value as defined in Section 1.7.2. This may be considered as paths using edges within two adjacent layers[2]. Visually, stable paths be represented as perpendicular to interesting paths. While stable paths do not consider interestingness, it may be useful to consider their relation to interesting paths computationally [1, 2].

2 Graph layers are defined in Section 1.8.4

3

SOLVING THE MAX-IP PROBLEM

The Max-IP is the problem of finding the path of maximum interestingness in a given graph. Note, interestingness is defined in Section 1.9. If the $\log_2$ factor is removed from the calculation of the interestingness score, the problem of finding the path of maximum interestingness score is equivalent to the problem of finding the longest path in a weighted directed graph. The initial formulation of the Max-IP problem is found in [41].

This section reviews the Max-IP problem, starting with its definition and motivation, followed by its NP-completeness in the general digraphs and its polynomial-time algorithm for directed acyclic graphs. We then argue why the output graph of the Mapper algorithm can be transformed to a directed acyclic graph in the case of the Max-IP problem.

3.1 PROBLEM

Consider the output graph G of the Mapper algorithm. Note that G is a directed graph with vertices V of size n and edges E of size m where each edge has a weight $w(e_i) \geq 0$. Find the path P such that its interestingness score $I(P)$ is maximized. Recall $I(P)$ is given by Equation 1.6: $I(P) = \Sigma_{r=1}^{k} w(e_r) \cdot \log_2(1+r)$, where $w(e_r)$ is the weight of the edge e_r and k is the length of the path.

3.2 MOTIVATION

Interestingness in the context of the Mapper algorithm denotes an amplification of a set of attributes over a given measure. The Max-IP problem, which may be seen as the search for the longest tail-heavy path, is motivated by practice. As tail weight tends to show the extent of information in a given direction, the maximum interesting path demonstrates the longest string of information.

Specifically, the path of maximum interestingness implies the set of attributes that has the longest sequence of amplifications over the clustering function g. This can be seen as the most expected longest sequence.

For example, in the world of recorded chess competitions, consider the Modern Line of the Sicilian defence of Najdorf. The most played opening is King's Pawn (1 move from white). Assuming King's Pawn is played, the most played opening is the Sicilian Defence. Assuming the Sicilian Defence is played, the most played opening is the Open Sicilian

Defence. Assuming the Open Sicilian Defence is played, the most played opening is the Sicilian Defence, Najdorf Variation. Assuming the Sicilian Defence, Najdorf Variation is played, the most played opening is the Modern Line of the Najdorf Variation [51].

The sequence of moves that lead to the Modern Line of the Najdorf Variation is the Max-IP solution to the graph of all recorded competitive chess openings, where the function g is the number of moves played, and the weights of edges are proportional to the recorded number of games with that move. In practice, the most popular lines are often the most studied.

Mathematically, the solution of the Max-IP can be understood as the most represented monotonic trend of the dataset. For example, in medicine or A/B testing, this may correspond to the control group, as opposite to the test group. The Mapper algorithm is intended for data mining of the flares and other topological features. Since flares are by definition deviations from the main path, the Max-IP problem does not correspond to the intention of the Mapper algorithm. However, in [41], an algorithm to find the Max-IP is used greedily to approximate the IP problem, seen in Section 5.5. This application motivates the use of solving the Max-IP. Other applications include the topological analysis of the graph G when all the edges of the maximum interestingness score path are removed. The latter creates a "flare" graph, that is, all paths in the graph deviate from the main path by definition, thus every path in the graph may be considered a flare.

3.3 GENERAL DIGRAPHS

Since the interestingness score of a path increases with the number of edges, for an unweighted general digraph, the Max-IP is equivalent to the longest path. Thus, the Max-IP can be proven [41] to be NP-complete with a reduction to the directed Hamiltonian Path problem (DirHC), which is one of the 21 NP-complete problems originally introduced by Karp [45].

3.4 DIRECTED ACYCLIC GRAPHS

In the case of a directed acyclic graph $G = (V, E)$ where $|V| = n$ and $|E| = m$, the Max-IP problem is in P. This is proven by the dynamic programming and backtracking algorithm described in [41], where the dynamic programming table is constructed such that the edges considered are on one axis, and the length of the path on the other axis. We call the following the Max IP algorithm for simplicity. Let $T(i, j)$ denote the interestingness score of a maximum interestingness score path of length j edges ending at edge e_i for $i \in 1, 2, ..., m$. The following algorithmic steps and Figure 3.1 are taken from [41]:

1. **Initialization:** $T(i, 1) = w(e_i) \log_2(2)$, where $1 \leq i \leq m$.

2. **Recurrence:** For an edge $e = (u, v) \in E$, we define a predecessor edge of e as any edge $e' \in E$ of the form $e' = (w, u)$. Let $\text{Pred}(e)$ denote the set of all predecessor edges of e. Note $\text{Pred}(e)$ may be empty. We define the recurrence for $T(i, j)$ as follows:

$$T(i,j) = \max_{e_{i'} \in Pred(e_i)} \{T(i', j-1) + w(e_i) \cdot \log_2(1+j)\} \quad (3.1)$$

3. **Output:** We report the score that is maximum in the entire table. A corresponding optimal path P is obtained by backtracking from that cell to the first column.

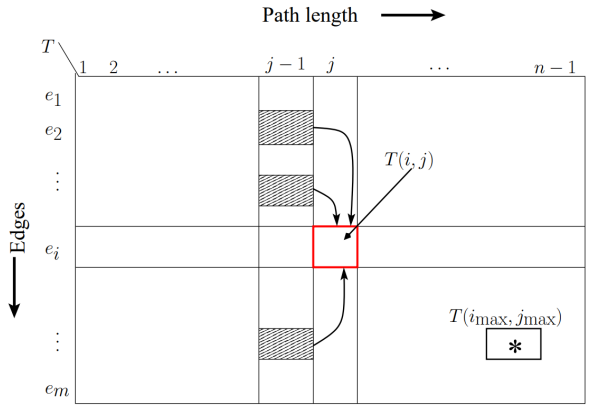


Figure 3.1: Table T(i, j) for the Max-IP algorithm

This method is proven to have a time complexity of $O(mnd_{in})$, where d_{in} is the maximum indegree of any vertex in G. Moreover, Kalyanaraman, Kamruzzaman and Krishnamoorthy [41] show this can be improved up to $O(m\delta_{max}d_{in})$, where δ_{max} is the graph diameter of G.

3.5 TRANSFORMING GENERAL TO DAG

The difficulty of the Max-IP is due to the difficulty of finding the specific path optimization in cyclic graphs. However, the output graph of the Mapper algorithm may be considered a multilayered graph. Considering those particularities, a general Mapper output may be transformed to a directed acyclic graph in polynomial time.

Theorem 3.5.1. *Consider a multilayered weight-balanced digraph G, where the tolerance $\tau \leq l/4$ and the overlap size $p < l/4$, where l is the length of the intervals that formed the layers[1] of G. Then the Max-IP problem can be solved in polynomial time on G.*

Proof. Consider a multilayered digraph G with tolerance $\tau \leq l/4$ and overlap $p < l/4$. The tolerance implies all bidirectional edges are isolated: there exists no two consecutive bidirectional edges. To

1 The terms tolerance, overlap, intervals, and layers are explained in Section 1.7.2.

prove this, a proof by contradiction is used, where two consecutive edges are bidirectional and $\tau < l/2$. By the graph construction, two edges cover 3 points on 3 different layers. This implies the g value of those edges is between $[0, l]$ and $[2l - 2p, 3l - 2p]$. Considering G is multilayered, $p < l/4$, which implies the following total weight w_e for 2 edges: $w_e > 2l - 2p - l > l - 2l/4 = l/2$. Considering these edges are bidirectional and consecutive, the maximum distance covered is $w_e \leq 2\tau \leq l/2$. This is a contradiction.

Now that bidirectional edges are isolated, notice these are the only cycles found in G. For this reason, removing these cycles enables the user to run the Max-IP on the graph in polynomial time. Consider the isolated cycle in Figure 3.2.

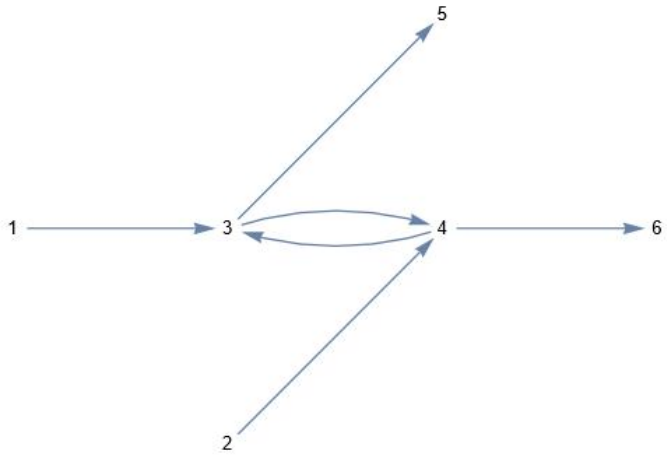


Figure 3.2: Graph representing an isolated bidirectional edge $\{v_3, v_4\}$

This subgraph may be changed to an equivalent acyclic subgraph seen in Figure 3.3. Note the red edges are "empty" edges, that is,

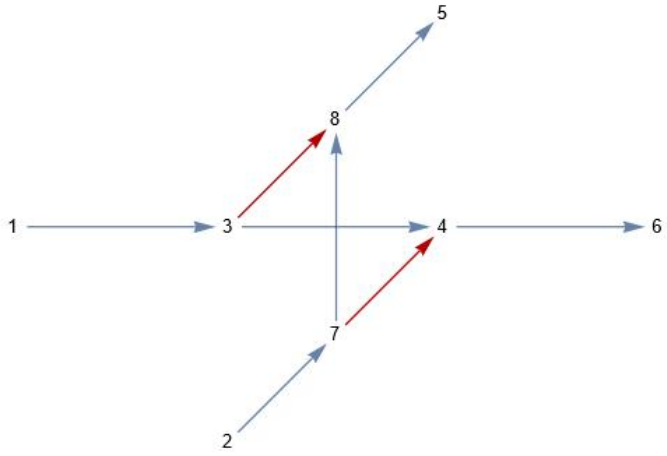


Figure 3.3: Graph representing the acyclic version of the bidirectional edge

edges with no weight and no impact on rank. This is done to ensure the length of the paths $(1, 3, 5)$ and $(2, 4, 6)$ maintain the same rank for

any path in G. This graph transformation allows for one path from v_1 to v_5, one path from v_1 to v_6, one path from v_2 to v_6 and one path from v_2 to v_5. The edge $\{v_2, v_4\}$ is represented by $\{v_2, v_7\}$, the edge $\{v_3, v_5\}$ is represented by $\{v_8, v_5\}$ and the edge $\{v_4, v_3\}$ is represented by $\{v_7, v_8\}$. Thus, all paths available for the initial subgraph are respected in the new subgraph. Without loss of generality, this is applicable to any number of edges incoming v_3 or outgoing v_4.

Therefore, consider a multilayered weight-balanced digraph G with the tolerance $\tau \leq l/4$ and overlap $p < l/4$. Then there exists a directed graph G' such that the cycles are removed. From there, the Max-IP algorithm for directed acyclic graphs may be used, with red edges having a weight of 0. □

Therefore, with this transformation, multilayered weight-balanced digraph G with the tolerance $\tau \leq l/4$ and overlap $p < l/4$ are solvable with the Max-IP algorithm even with cycles.

4

THE K-IP PROBLEM

The k-IP problem, if solved, allows the user to manually adjust the length of the flares considered. Flares are defined in Section 1.9. This is useful for the analysis of datasets where the sample size needed for significance is known. This allows for a stronger graph analysis based on significant flare decomposition. Note, interestingness is defined in Section 1.9.

This section explores the k-IP problem, starting with its definition and motivation, followed by an explanation of why its proof of NP-completeness in [41] has gaps. Then, we present three approximation algorithms and three exact algorithms to allow the practical use of length-bound interesting paths. Finally, the experimental results of these algorithms are briefly mentioned and discussed.

4.1 PROBLEM

Consider the directed graph $G = (V, E)$, with vertices V of size n and edges E of size m where each edge has a weight $w(e) \geq 0$. The k-IP problem is to find the edge-disjoint set P of k-paths (paths of length exactly k) such that the sum of their interestingness score is maximized.

This problem is naturally formulated as an optimization problem. In the interest of complexity analysis, the decision version of k-IP, termed k-IPD, is defined as follows. Consider the output graph G of the Mapper algorithm, with vertices V of size n, edges E of size m and a target score of $t_0 \geq 0$. The goal is to determine whether there exists an edge-disjoint set P of k-paths (paths of length exactly k) such that the sum of their interestingness score is greater or equal to a target weight t_0.

4.2 MOTIVATION

By restricting the problem to flares of exactly size k, the k-IP problem avoids large flares that are too general and smaller insignificant flares. In practice [42], flares of the Mapper output are characterized by their significant interestingness and deviation from common paths. The maximum path found in the Max-IP problem leads to a large interestingness score, but it does not deviate from common paths, unlike flares. On the other hand, paths of non-significant interestingness score may easily avoid common paths, but their smaller size renders their analysis difficult and less interesting. Thus, data exploration of

the Mapper output requires a set of paths of significant size, which is by definition near-linearly correlated to the interestingness score. Thus, the formulation of the k-IP problem aims to allow efficient subsequent iterative flare analysis. Considering edge weights are approximately uniform 1.8.5, the notion of path significance can be attributed to a given length k. As a simplifying assumption of the problem of interestingness score maximization, this problem restricts the length of any path to be exactly k, thus resulting in a relatively uniform interestingness score among all paths. Moreover, for the data analysis to be exhaustive, the set of interesting paths has to be optimal by some metric. If we define the weight of a set of paths as the sum of the weights of each path, then optimizing for the highest weight naturally makes the set optimal. The goal of the k-IP problem is to facilitate the iterative analysis of significant k-paths that significantly deviate from the main path (given by the optimal solution to the Max-IP), to identify significant subpopulations of interest.

The following sections consider diverse simplifying assumptions, which are motivated and explained in Section 1.8. In the general case, the reader should assume graphs are weighted and directed multigraphs unless otherwise mentioned.

4.3 THE CASE k=2

A path has a positive length by definition, where path length is the number of edges it contains. The 1-IP problem is trivial, as any solution corresponds to the set E, since paths of length 1 are edges, with a total weight of $|E| \log_2(2) = |E|$. Consider therefore the 2-IP problem.

Theorem 4.3.1. *For weighted directed graphs, the 2-IP problem is in P.*

This proof is given in [41], and repeated here for references.

Proof. Consider an input $G = (V, E)$ to the 2-IP problem. We write edges in E as $u \to v$. We define the undirected graph $G' = (V', E')$, where $V' = E$ and $\{u \to v, u' \to v'\}$ is an edge in E' if and only if $v = u'$ or $v' = u$.

This graph construction implies each vertex in G' corresponds to an edge in E and each edge in G' corresponds to an interesting 2-path in G. This can be seen in Figures 4.1 and 4.2.

The weight of each edge $e_{i,j} = \{e_i, e_j\} \in E'$ is the maximum of the interestingness score of the paths $\{e_i, e_j\}$ and $\{e_j, e_i\}$: $w_{e_{i,j}} = \max\{I(\{e_i, e_j\}), I(\{e_j, e_i\})\}$, where $I(p)$ is the interestingness score of p. This implies the maximum number of different weights in E' is $\binom{m}{2} = m(m-1)/2$. Since this results in a polynomial number of maximum-weight 2-paths over G, this maximum weight matching in G' corresponds to an optimal solution of the 2-IP in G. Consider the maximum weight matching problem on an undirected graph with n

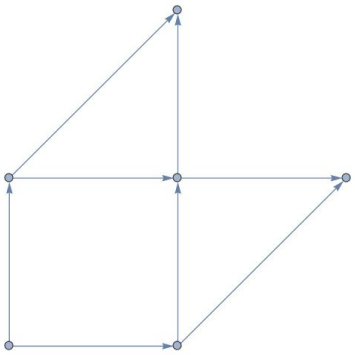


Figure 4.1: Graph G in blue

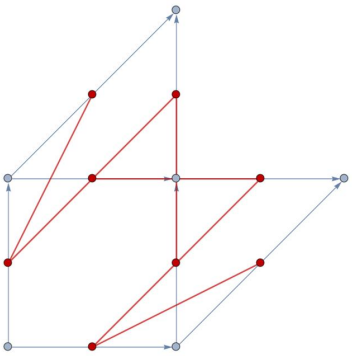


Figure 4.2: Graph G in blue and graph G' in red

vertices and m edges can be solved with Gabow's[24] implementation of Edmonds' algorithm in $O(nm + n^2 \log_2 n)$ time. Thus, we can compute a maximum weight matching in G' in $O(m'^3)$ time. In terms of the input graph G, the 2-IP problem can be solved in $O(m^6)$ time. Hence, the k-IP problem is in P for $k \leq 2$. □

4.4 NP-COMPLETENESS OF THE CASE k=3

To show NP-completeness of the 3-IP problem, Kalyanaraman, Kamruzzaman and Krishnamoorthy [41] provide a reduction from the exact 3-cover (3XC) problem. The input to the 3XC problem consists of a set $X = \{x_1, x_2, ..., x_{3q}\}$ and a collection $\mathscr{C} = \{t_1, t_2, ..., t_p\}$ of sets of exactly 3 distinct elements of X. The desired output is a set $\mathscr{C}' \subseteq \mathscr{C}$ such that $|\mathscr{C}'| = q$ and for any t_i and t_j in $\mathscr{C}'$, $t_i \cap t_j = \emptyset$. This problem is a special case of the exact cover problem, one of Karp's original 21 problems, and was proven to be NP-complete in [31].

We found the proof in [41] of the NP-completeness of the 3-IP problem to have gaps. The following subsections review and explain in detail the graph construction used in [41]; then, we demonstrate the gap in the proof.

4.4.1 *Graph construction*

In this section, we provide the construction of [41], a directed acyclic graph $G = (V, E)$ from a given instance of the 3XC$(X, \mathscr{C})$ problem.

Without loss of generality, all elements in X are ordered from 1 to n. Using this arbitrary ordering of the elements of X, on which each subset $t_i \in \mathscr{C}$ is locally sorted, we consider all triples t_i ordered. Consider an arbitrary subset $t_i = (x, y, z)$. Each graph Γ_i represents a triple t_i using the graph construction shown in Figure 4.3. All graphs $\{\Gamma_1, \Gamma_2, ..., \Gamma_p\}$ of all other subsets $t_j \in \mathscr{C}$ are isomorphic to Γ_i. Vertices are denoted $v_{i,q}$, where i denotes the subgraph Γ_i and $q \in [1, 13]$. Any

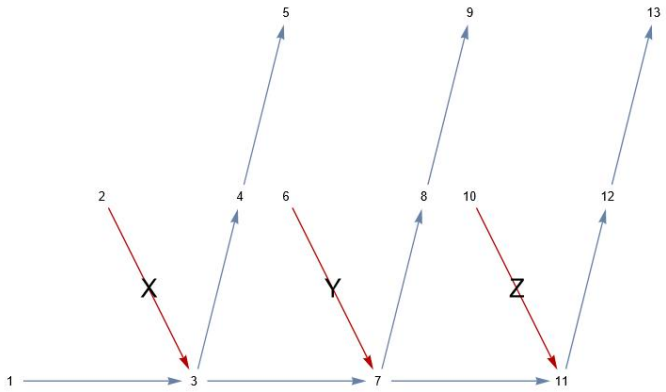


Figure 4.3: A subgraph Γ_i representing the set of elements $t_i = (X, Y, Z)$.

edge may be denoted as $(v_{i,q}, v_{i,r})$ or in short form as $e_{i,q,r}$, where i denotes the subgraph Γ_i, q denotes the topological position of its first vertex and r denotes the topological position of its second vertex in Γ_i. Thus, q and r are integers between 1 and 13. We define a named edge as an edge of the form $e_{i,2,3}$, $e_{i,6,7}$ or $e_{i,10,11}$. In Figure 4.3, named edges are shown in red. Moreover, each named edge has a unique correspondence to an element of X, indicated by its name. For named edges, we add the additional notation ξ_i, where i is the element in X this edge is assigned to. The first and second vertices of a named edge are referred to as $v_{\xi_i,1}$ and $v_{\xi_i,2}$ respectively.

The graph $G = (V, E)$ is constructed in the following manner:

$$V = (\cup_{i=1}^{p}\{v_{i,1}, v_{i,4}, v_{i,5}, v_{i,8}, v_{i,9}, v_{i,12}, v_{i,13}\}) \cup (\cup_{j=1}^{3q}\{v_{\xi_j,1}, v_{\xi_j,2}\}) \quad (4.1)$$

$$E = (\cup_{i=1}^{p}\{e_{i,1,3}, e_{i,3,4}, e_{i,4,5}, e_{i,3,7}, e_{i,7,8}, e_{i,8,9}, e_{i,7,11}, e_{i,11,12}, e_{i,12,13}\}) \cup \{\xi_j | j \in [1, 3q]\} \quad (4.2)$$

All unnamed edges (blue) have a weight of 1 and all named (red) edges have a weight of p, where p is the number of triples in $\mathscr{C}$. Note, this construction implies all unnamed (blue) edges $\{e_{i,1,3}, e_{i,3,4}, e_{i,4,5}, e_{i,3,7}, e_{i,7,8}, e_{i,8,9}, e_{i,7,11}, e_{i,11,12}, e_{i,12,13}\}$ in Figure 4.3 are each associated to exactly 1 subgraph, however all named edges ξ_j as well as their vertices $\{v_{\xi_j,1}, v_{\xi_j,2}\}$ are unique and may be shared among multiple graphs Γ. This allows multiple notations for named edges: for example, in Figure 4.3, ξ_X is equivalent to $e_{i,2,3}$.

4.4.1.1 *Examples*

Example 4.4.1. *While each subgraph is a simple graph, G is a multigraph. Consider the 3XC problem where* $X_1 = \{A, B, C, D\}$ *and* $\mathscr{C}_1 = \{t_1, t_2\} = \{(A, B, C), (A, B, D)\}$. *This can be shown in Figure 4.4, where the edges* $e_{1,3,7}$ *and* $e_{2,3,7}$ *share the same vertices.*

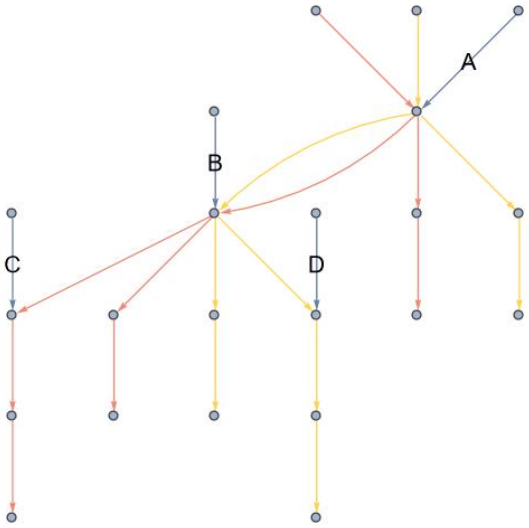


Figure 4.4: Example of the graph G for $\mathscr{C}_1$, where the edges $e_{1,3,7}$ and $e_{2,3,7}$ share the same vertices

Example 4.4.2. *Consider the 3XC problem where* $X_2 = \{A, B, C, D, E, F\}$ *and* $\mathscr{C}_2 = \{(A, B, C), (A, C, D), (B, D, F), (B, E, F)\}$. *This is shown in Figure 4.5.*

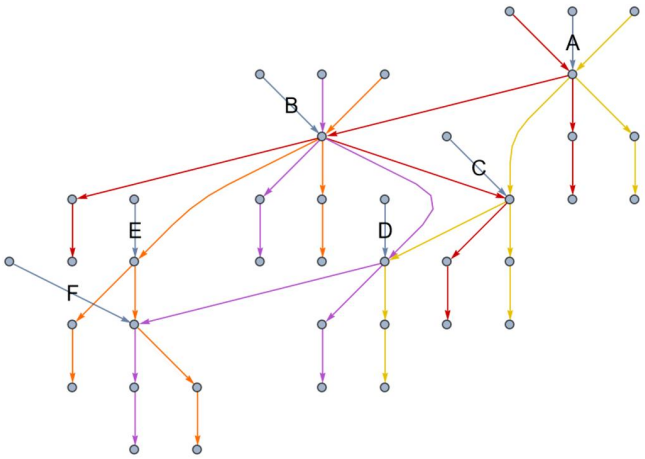


Figure 4.5: Example of the graph G for $\mathscr{C}_2$

All subgraphs are isomorphic and edges are coloured according to their subgraph Γ_i, *corresponding to* t_i. *Notice, edges that represent an element are not coloured, since such edges can be shared by multiple sets.*

Example 4.4.3. *Consider the 3XC problem where* $X_3 = \{A, B, C, D, E, F, G, H, I\}$ *and* $\mathscr{C}_3 = \{(A, B, C), (A, C, D), (B, D, F), (B, E, F), (E, G, H), (A, E, I), (C, G, H)\}$. *The corresponding graph G is given in Figure 4.6.*

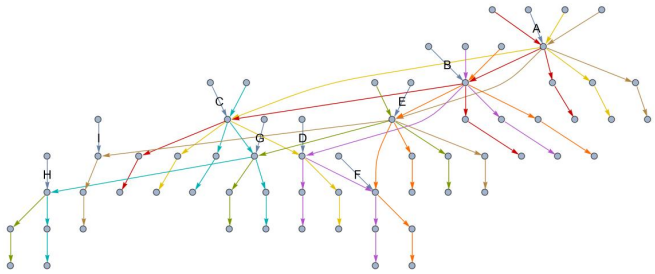


Figure 4.6: Example of the graph G for $\mathscr{C}_3$

4.4.1.2 *Graph Properties*

Lemma 4.4.4. *G is acyclic.*

Proof. Each of the vertices $v_{i,1}$, $v_{i,2}$, $v_{i,5}$, $v_{i,6}$, $v_{i,9}$, $v_{i,10}$, and $v_{i,13}$ has in- or out-degree zero. Therefore, they cannot be part of any cycle. The vertices $\{v_{i,4}, v_{i,8}, v_{i,12}\}$ form unique and disconnected paths for each subgraph. Therefore, no cycle can be formed using their edges. All vertices $\{v_{i,3}, v_{i,7}, v_{i,11}\}$ have edges that connect from one element x_i in X to another element x_j in X, the latter always being ranked lower. Thus, a topological sorting is possible based on the sorting of X, which is equivalent to stating G is acyclic. □

Consider the following property of the topology of Γ_i.

Lemma 4.4.5. *Any maximum partition of Γ_i has exactly four 3-paths.*

Proof. Consider the 1^{st} maximum 3-IP partition of Γ_i, given in Figure 4.7. Note, there is exactly one other configuration that accomplishes

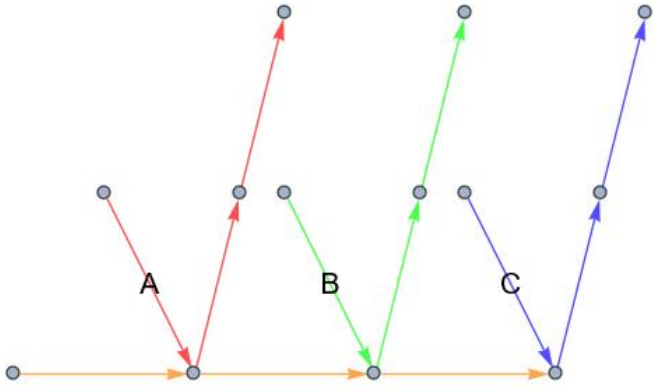


Figure 4.7: 1^{st} Maximum 3-IP Subgraph Partition

the same result, seen in Figure 4.8. Since all edges are used, there is no greater 3-path partition. In both cases, the total number of 3-paths of this subgraph is four. □

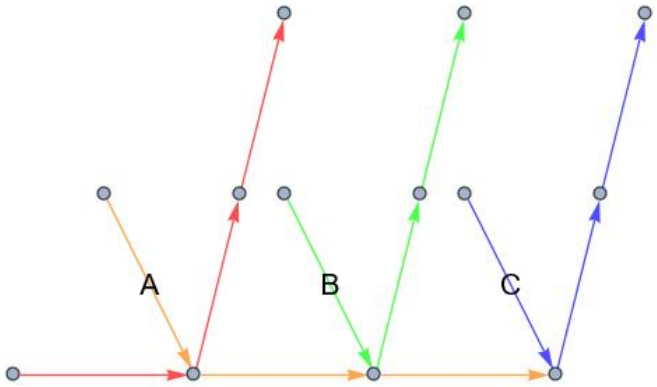


Figure 4.8: 2^{nd} Maximum 3-IP Subgraph Partition

We note here the total interestingness score of these four paths. Since there are 3 named edges, and in all cases these edges are the first edge of the path, the total interestingness score is:

$$W_{in} = 3(p\log_2 2 + \log_2 3 + \log_2 4) + (\log_2 2 + \log_2 3 + \log_2 4) \tag{4.3}$$

$$= 4\log_2 24 + 3(p-1) \tag{4.4}$$

Lemma 4.4.6. *If any positive number of the named edges ξ_j are removed from Γ_i, the resulting graph Γ_i' can always be partitioned into three 3-paths.*

Proof. This is shown by the fact the resulting graph has between 9 and 11 edges. Thus, it cannot have more than three 3-paths. If all named edges ξ_i are removed from the graph Γ_i, then it has three 3-paths, as shown in Figure 4.9. □

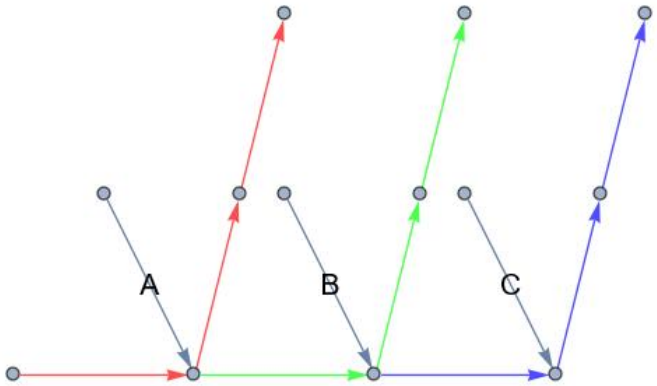


Figure 4.9: Example: a maximum partition subgraph without using 3 named edges

We note here the total interestingness score of these three 3-paths, seen in Figure 4.9:

$$W_{out} = 3(\log_2 2 + \log_2 3 + \log_2 4)$$

$$= 3\log_2 24 \tag{4.5}$$

4.4.2 *Reduction: from 3XC to 3-IP*

Lemma 4.4.7. *If the 3XC problem of $\mathscr{C}$ on X has an exact cover, then the 3-IP problem of G has a total interestingness score of $t_0 = 3(p+q)\log_2 24 + 3q(p-1)$.*

Proof. Without loss of generality, consider $\mathscr{C}' = \{(x_1, x_2, x_3), (x_4, x_5, x_6), ..., (x_{3q-2}, x_{3q-1}, x_{3q})\}$. For each set of $\mathscr{C}$ included in $\mathscr{C}'$, cover the corresponding subgraph Γ_i by q sets of four 3-paths using the construction in Lemma 4.4.5. This is possible since all elements in $\mathscr{C}'$ are pairwise disjoint. Thus, all corresponding named edges are also pairwise disjoint. For all other sets, cover the corresponding graph Γ_i by $p-q$ sets of three 3-paths using the construction in Figure 4.9. This is possible since in the equation 4.2 of the graph construction, each set of unnamed edges is disjoint from any other set of unnamed edges. Considering $\mathscr{C}'$ covers exactly the set X, this implies all edges of G are used. Thus, the total interestingness score of $\mathscr{C}'$ is the total interestingness score of q subgraphs with named edges and $p-q$ subgraphs without the named edges. The total interestingness score is given by Equations 4.4 and 4.5:

$$\begin{aligned} t_0 &= qW_{in} + (p-q)W_{out} \\ &= (3p-q)\log_2 24 + 3q(p-1) \end{aligned} \tag{4.6}$$

□

4.4.2.1 *Solutions to Examples*

Solution 4.4.8. *There is no solution to Example 1 as the size of X_1 is not divisible by 3.*

Solution 4.4.9. *Consider the 3XC problem where $X_2 = \{A, B, C, D, E, F\}$ and $\mathscr{C}_2 = \{(A, B, C), (A, C, D), (B, D, F), (B, E, F)\}$.*

A possible solution for this problem is the partition:

$$\mathscr{C}_2' = \{(A, C, D), (B, E, F)\} \tag{4.7}$$

This results in four 3-paths for two sets using the Figure 4.7 construction, and three 3-paths for two sets using the Figure 4.9 construction. This is given in Figure 4.10. Note, all subgraphs are isomorphic and edges are coloured according to their subgraph and named edges are coloured according to $\mathscr{C}_2'$.

Solution 4.4.10. *Consider the 3XC problem where $X_3 = \{A, B, C, D, E, F, G, H, I\}$ and $\mathscr{C}_3 = \{(A, B, C), (A, C, D), (B, D, F), (B, E, F), (E, G, H), (A, E, I), (C, G, H)\}$. For this example, a possible solution is the partition:*

$$\mathscr{C}_3 = \{(B, D, F), (A, E, I), (C, G, H)\} \tag{4.8}$$

This results in four 3-paths for 3 sets using the Figure 4.7 construction, and three 3-paths for four sets using the Figure 4.9 construction. This is given in Figure 4.11.

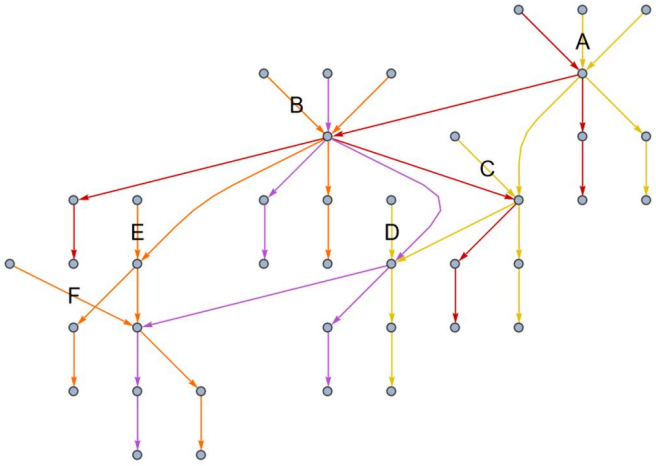


Figure 4.10: Example: an optimal solution for $\mathscr{C}_1$

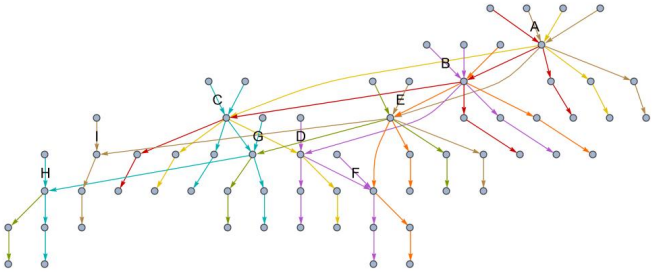


Figure 4.11: Example: an optimal solution for $\mathscr{C}_3$

4.4.3 *Proof of gaps*

In [41], it is claimed that if the interestingness score of G is greater than or equal to t_0, then X has an exact cover. In Lemma 4.4.11, we provide a counter example.

Lemma 4.4.11. *If the 3-IP problem of G has a total interestingness score t_0, it does not imply the 3XC problem of $\mathscr{C}$ on X has an exact cover.*

Proof. Consider the 3XC problem where $X_4 = \{A, B, C, D, E, F, G, H, I\}$ and $\mathscr{C}_4 = \{(A, B, C), (C, D, G), (E, F, G), (A, H, I)\}$. The corresponding graph G_4 is shown in Figure 4.12.

All subgraphs are isomorphic and edges are coloured according to their subgraph. Notice, edges that represent an element are not coloured, since such edges can be shared by multiple tuples.

Lemma 4.4.12. *There is no possible exact cover for $G_4 = (X_4, \mathscr{C}_4)$.*

Proof. Consider the tuple (A, B, C). Either it is included or excluded in the cover.

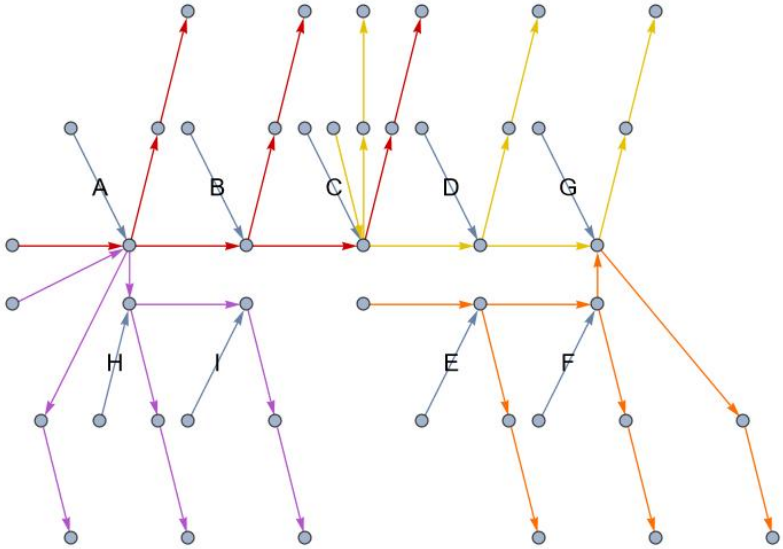


Figure 4.12: The graph G_4 for the collection $\mathscr{C}_4$ of Lemma 4.4.11

If (A, B, C) is included, (A, H, I) is excluded as it intersects on A. However, its exclusion implies H is excluded, since it is not part of any other set.

If (A, B, C) is excluded, its exclusion implies B is excluded, since it is not part of any other set. □

However, there is a solution with total interestingness score t_0. This path decomposition of G_4 in fifteen 3-paths is given in Figure 4.13.

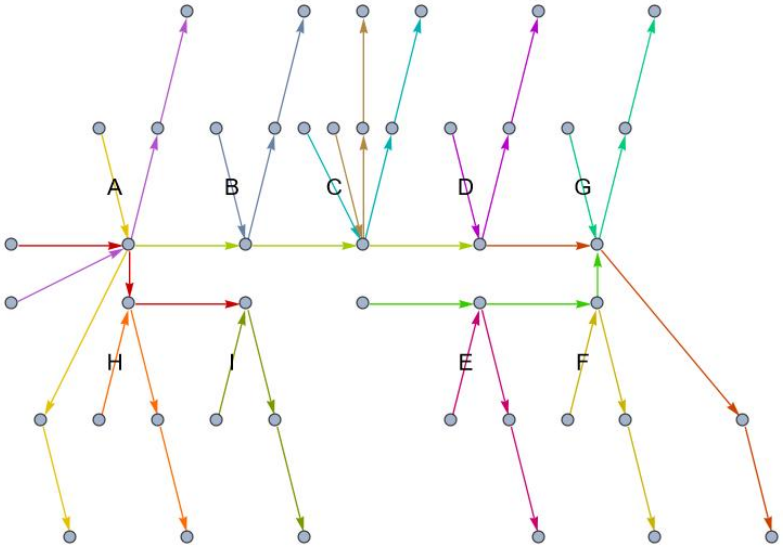


Figure 4.13: An exact 3-path partition of G_4

This path partition is composed of nine 3-paths where each first edge is named and six 3-paths composed of unnamed edges. Consider

$p = |\mathscr{C}_4| = 4$ and $q = |X_4|/3 = 3$. This has a total interestingness score of:

$$I(P) = 9(p + \log_2(3) + \log_2(4)) + 6\log_2(24) \tag{4.9}$$
$$= 72 + 15\log_2(3) \tag{4.10}$$

The value t_0 is equal to this value:

$$t_0 = qW_{in} + (p-q)W_{out} \tag{4.11}$$
$$= 3(p-1)q + (3p+q)\log_2(24) \tag{4.12}$$
$$= 3\log_2(3) + 36\log_2(2) + 12(\log_2(3) + \log_2(8)) \tag{4.13}$$
$$= 72 + 15\log_2(3) \tag{4.14}$$

This is a contradiction, as the maximum solution for the 3-IP problem of G_4 has an interestingness score greater or equal to t_0 but the 3XC problem of $\mathscr{C}$ on X does not have an exact cover. □

4.4.4 *Conclusion*

This counterexample demonstrates the NP-completeness reduction presented in [41] has gaps, and thus, the hardness of the k-IP problem for $k = 3$ is open.

4.5 NP-COMPLETENESS OF THE CASE K>3

This section provides a proof of the NP-completeness of the k-IP problem for unweighted directed acyclic graphs. To achieve this, we first motivate the use of an equivalent problem, the DAED4 problem, which we then define. The proof of the NP-completeness of the DAED4 problem is given for unweighted directed acyclic graphs. This leads to the NP-completeness of the DAEDk for $k > 3$ on unweighted directed acyclic graphs, which then leads to the NP-completeness of the k-IP for $k > 3$ on unweighted directed acyclic graphs.

4.5.1 *Definitions*

4.5.1.1 *Directed Edge-Disjoint k-Path Partition problem*

Closely related problems to the k-IP problem include the k-path partition problem [37], the unsplitable flow problem [21] and other variants considered in Section 2.3. We first extend the edge-disjoint exactly-k paths problem to directed graphs.

The Directed Edge-Disjoint k-Path Partition problem is defined as follows. Consider a directed graph $G = (V, E)$ such that $|V| = n$ and $|E| = m$. Consider a set of start vertices $\{s_1, s_2, s_3, \ldots, s_u\} \subset V$ and a corresponding set of end vertices $\{t_1, t_2, t_3, \ldots, t_u\} \subset V$. Find the

pairwise edge-disjoint set P of k-paths (paths of exactly k connected edges) such that every path p_i starts at s_i and ends at t_i [37].

The directed edge-disjoint k-path partition problem is shown to be NP-complete for $k \geq 3$ [21].

4.5.1.2 *Directed Acyclic Edge-Disjoint k-Path Partition (DAEDk) problem*

However, in the unweighted k-IP problem on directed acyclic graphs, the start and end vertices are not specified, and graph cycles are not allowed. We consider a new variant of the Directed Edge-Disjoint k-Path Partition problem on directed acyclic graphs with unspecified start and end vertices (DAEDk). Once its NP-completeness is proven, it can be used to prove the NP-completeness of the unweighted k-IP problem.

We define the DAEDk problem as follows. Consider an unweighted directed acyclic graph $G = (V, E)$ such that $|V| = n$ and $|E| = m$. Determine whether there exists a pairwise edge-disjoint set P of k-paths (paths of exactly k connected edges) such that the number of k-paths in P is greater or equal to a given integer y.

Theorem 4.5.1. *Let $G = (V, E)$ be an unweighted directed graph. The set of paths P to solve the DAEDk problem on G is optimal if and only if P is optimal in the unweighted k-IP problem on G.*

Proof. Consider the solution P of the DAEDk on G. This implies the maximum set of k-path on G. Thus, this yields the maximum interestingness score on G, since the maximum interestingness score is based on the number of k-paths exclusively.

Consider the solution P of the k-IP problem on G. This implies the maximum interestingness score, which is based on the number of k-paths exclusively. Thus, this yields the maximum number of k-paths, which solves the DAEDk problem on G. □

4.5.1.3 *The decision DAED4 problem*

Consider an unweighted directed acyclic graph $G = (V, E)$ such that $|V| = n$ and $|E| = m$ and consider an integer OPT. Determine whether there exists a pairwise edge-disjoint set $\mathscr{C}'$ of 4-paths (paths of exactly 4 connected edges) such that the number of 4-paths in $\mathscr{C}'$ is greater or equal to OPT.

4.5.1.4 *The decision SAT problem*

Consider the following general SAT formulation. Given a Boolean formula $B = \gamma_1 \wedge \gamma_2 \wedge \cdots \wedge \gamma_m$ on n Boolean variables $X = \{x_1, x_2, \cdots, x_n\}$ which is the conjunction of a set of m disjunctive clauses $C = \{\gamma_1, \gamma_2, \cdots, \gamma_m\}$, where each literal of each clause is either x_i or $\bar{x}_i$, determine whether there exists a truth assignment to its variables for which the formula B is True.

Without loss of generality, we can assume that no clause has a variable and its negation, as that clause is always True. Using a search algorithm over all clauses with a given literal and its negation, all such clauses are removed in polynomial time.

4.5.1.5 *Graph union*

We define graph union as follows: the union $G = G_i \cup G_j$ of graphs G_i and G_j with disjoint vertex sets V_i and V_j respectively and edge sets E_i and E_j respectively is the graph G with $V = V_i \cup V_j$ and $E = E_i \cup E_j$. [28, 30]

4.5.2 *Certificate of polynomial length*

Lemma 4.5.2. *The DAED4 problem for unweighted directed acyclic graphs is in NP.*

Proof. Consider a collection $\mathscr{C}'$ of 4-paths in an unweighted directed acyclic graph $G = (V, E)$ where $|E| = p$ with a target weight of t_0. In polynomial time, paths can be verified pairwise edge-disjoint. In $O(1)$ time, the size of $\mathscr{C}'$ can be evaluated. Thus, the DAED4 problem is in NP. □

4.5.3 *Graph construction*

In this section, we provide a construction of a directed acyclic graph $H = (V, E)$ from a given instance of a conjunction of m disjunctive clauses including n Boolean variables $X = \{x_1, x_2, \cdots, x_n\}$, from the SAT problem.

We create unweighted directed acyclic graphs X_i for each variable x_i. We denote the vertices of the graph X_i as $v_{i,j,k}$, where i is for the variable x_i, j is the position of the clause in B where x_i occurs and $k \in \{1, 2, 3, 4, 5, 6\}$. The edges in X_i are the following: $(v_{i,j,1}, v_{i,j,2})$, $(v_{i,j,2}, v_{i,j,3})$, $(v_{i,j,3}, v_{i,j-1,5})$, $(v_{i,j,2}, v_{i,j,4})$, $(v_{i,j,4}, v_{i,j,5})$ and $(v_{i,j,5}, v_{i,j,6})$, where i is for the variable x_i, j is the position of the clause where x_i occurs. Note $j - 1$ refers to the edge connecting to the preceding clause containing x_i. In the case of the first occurrence of x_i, it refers to the last clause containing x_i. This is shown in Figure 4.14. Consider the graph $H_X = \cup_{i=1}^{n} X_i$ is the disjoint union of all X_i graphs. We add additional vertices and edges to the graph H_X named clause vertices and clause edges respectively. We denote the clause vertices as $c_{j,k}$ where j is the position of the clause γ_j in B and $k \in \{1, 2, 3\}$ corresponds to the k^{th} vertex associated to the clause γ_j. The clause edges are the following: $(c_{j,1}, c_{j,2})$, $(c_{j,2}, c_{j,3})$, and $(c_{j,3}, v_{i,j,3})$ or $(c_{j,3}, v_{g,j,4})$, where j is the position of the clause where x_i occurs, i is for the variable x_i that occurs in its positive form (x_i) in the clause j,

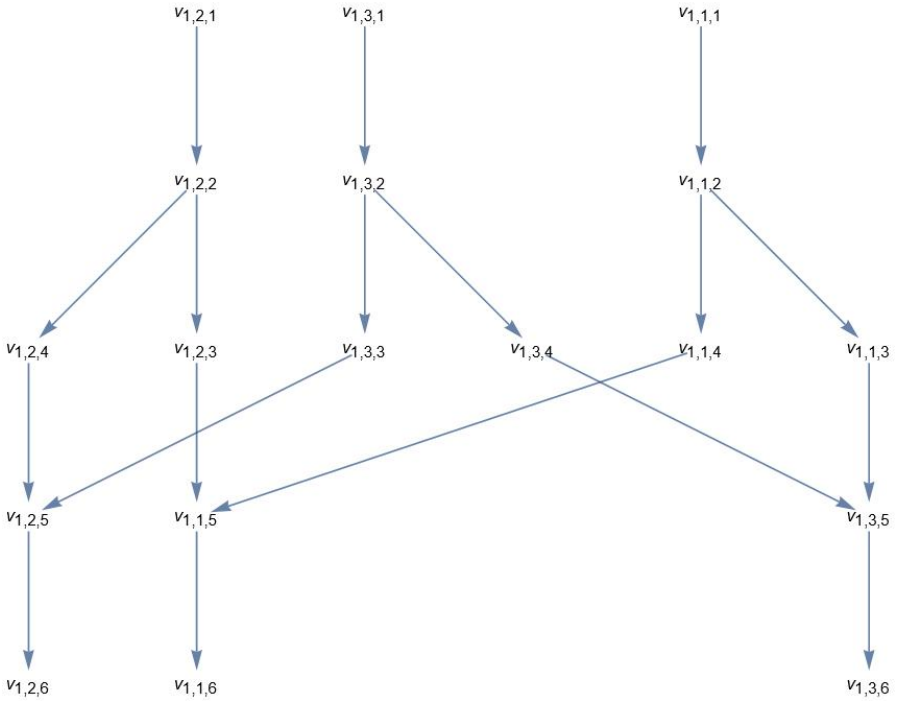


Figure 4.14: Example: Graph X_i where the variable x_i is part of clauses 1, 2 and 3

and g is for the variable x_g that occurs in its negative form ($\bar{x}_g$) in the clause j.

We construct the graph $H = (V, E)$ by adding all clause vertices and clause edges to H_X. This is described algorithmically in Algorithm 3.

4.5.4 *Reduction*

Theorem 4.5.3. *The DAED4 problem in the case of an unweighted directed acyclic graphs is NP-hard.*

We first prove the following.

Theorem 4.5.4. *Consider a Boolean expression B given in the previously mentioned SAT problem. The graph H can be partitioned in at least $m + \Sigma_{i=1}^{n} \beta_i$ 4-paths if and only if the Boolean expression B is satisfiable, where β_i is the number of occurrences of a variable x_i within B.*

Proof. First, consider the equation B is satisfiable by a given assignment of truth values x_i^* for each variable x_i within B.

We denote β_i the number of occurrences of a variable x_i in B. Each isolated graph X_i may be maximally path partitioned in β_i 4-paths, as there are only β_i edges (of the form $(v_{i,j,1}, v_{i,j,2})$) that may be used as the first edges of any 4-path and there are only β_i edges (of the form $(v_{i,j,5}, v_{i,j,6})$) that may be used as the last edges of any 4-path. We denote any 4-path starting with the edges $(v_{i,j,1}, v_{i,j,2})$, $(v_{i,j,2}, v_{i,j,3})$ as an

Algorithm 3 Graph construction of H

Require: An instance of SAT with Boolean expression $B = (C, X)$

```
H ← ∪_{i=1}^{n} X_i
for each clause γ_j ∈ C do
    Add the edges (c_{j,1}, c_{j,2}) and (c_{j,2}, c_{j,3}) to H
    for each variable x_i ∈ γ_j do
        if x_i in γ_j is positive then
            Add the edge (c_{j,3}, v_{i,j,3}) to H
        else                                   ▷ x_i in γ_j is negative()
            Add the edge (c_{j,3}, v_{i,j,4}) to H
        end if
    end for
end for
return H
```

upper path, any 4-path starting with the edges $(v_{i,j,1}, v_{i,j,2}), (v_{i,j,2}, v_{i,j,4})$ as a *lower path*, and any path starting with $(c_{j,1}, c_{j,2}), (c_{j,2}, c_{j,3})$ as a *clause path*. Note that if both a lower path and an upper path are used to partition the same graph X_i, it implies at least one of the $(v_{i,j,5}, v_{i,j,6})$ edges will not be in any disjoint 4-path of H. Thus, for there to be β_i disjoint 4-paths in X_i, either all 4-paths within X_i must be upper paths or all 4-paths within X_i must be lower paths.

We start with an empty path partition set S for H. For each variable x_i assigned to True, we add to S all lower paths within the subgraph X_i of H. For each variable x_i assigned to True, we add to S all lower paths within the subgraph X_i of H. Since the edges of all X_i graphs are disjoint, this implies every X_i graph generates β_i 4-paths. Then, since the expression B is satisfied, each clause γ_j must be True, which implies at least one variable x_i expression is True. In the case the variable expression satisfied is x_i, we add to S the corresponding clause path $\{(c_{j,1}, c_{j,2}), (c_{j,2}, c_{j,3}), (c_{j,3}, v_{j,i,3}), (v_{j,i,3}, v_{j-1,i,5})\}$.[1] In the case the variable expression satisfied is $\bar{x}_i$, we add to S the corresponding clause path $\{(c_{j,1}, c_{j,2}), (c_{j,2}, c_{j,3}), (c_{j,3}, v_{j,i,4}), (v_{j,i,4}, v_{j,i,5})\}$. Since the edges of all clause paths are disjoint, this implies every clause generates one 4-path. This gives a total of $m + \Sigma_{i=1}^{n} \beta_i$ 4-paths.

This proves the graph H can be partitioned in at least $m + \Sigma_{i=1}^{n} \beta_i$ 4-paths if the Boolean expression B is satisfiable. The following proves the converse.

Second, consider the graph H corresponding to the Boolean expression B has $m + \Sigma_{i=1}^{n} \beta_i$ disjoint 4-paths. Every 4-paths in H is part of exactly one out of four categories: upper paths, lower paths, clause paths and the last category of 4-paths which we define as follows. We denote as *middle paths*, 4-paths that start at an edge of the form

1 Note $j - 1$ refers to the edge connecting to the preceding clause containing x_i. In the case of the case of the first occurrence of x_i, it refers to the last clause containing x_i.

$(c_{j,2}, c_{j,3})$ and end at an edge of the form $(v_{j',i,5}, v_{j',i,6})$. Note all upper paths, lower paths and middle paths use the edges of the form $(v_{j,i,5}, v_{j,i,6})$. Moreover, note all clause paths and middle paths use edges of the form $(c_{j,2}, c_{j,3})$. However, there are only $\Sigma_{i=1}^{n}\beta_i$ edges of the form $(v_{j,i,5}, v_{j,i,6})$ and m edges of the form $(c_{j,2}, c_{j,3})$.

We claim if H has $m + \Sigma_{i=1}^{n}\beta_i$ 4-paths, then none of these paths are middle paths. By contradiction, if one of these paths is a middle path, then there remains only $m + \Sigma_{i=1}^{n}\beta_i - 2$ edges of the form $(v_{j,i,5}, v_{j,i,6})$ or $(c_{j,2}, c_{j,3})$ for the remaining $m + \Sigma_{i=1}^{n}\beta_i - 1$, each of which requires at least one of these edges of specific form. Thus, this is a contradiction.

Thus, the set of $m + \Sigma_{i=1}^{n}\beta_i$ 4-paths is composed of upper paths, lower paths and clause paths. This implies each subgraph X_i of H must contain β_i 4-paths. This implies, from previous remarks, that each subgraph contains either β_i lower paths and no upper paths or β_i upper paths and no lower paths. We consider that x_i is True if the 4-paths of X_i are lower paths, and that x_i is False if the 4-paths of X_i are upper paths. We denote this assignment of truth values as A.

In the case of a subgraph X_i of H where all disjoint 4-paths given are lower paths, this implies all edges of the form $(v_{j,i,3}, v_{j',i,5})$ are available and all edges of the form $(v_{j,i,4}, v_{j',i,5})$ are unavailable to clause paths containing these edges. The edge $(v_{j,i,3}, v_{j',i,5})$ corresponds to the last edge of the clause paths for the clause of B containing x_i as True, which corresponds to the assignment A.

Inversely, in the case of a subgraph X_i of H where all disjoint 4-paths given are upper paths, this implies all edges of the form $(v_{j,i,4}, v_{j',i,5})$ are available and all edges of the form $(v_{j,i,3}, v_{j',i,5})$ are unavailable to clause paths containing these edges. The edge $(v_{j,i,4}, v_{j',i,5})$ corresponds to the last edge of the clause paths for the clause of B containing x_i as False, which corresponds to the assignment A.

Since there are at most $\Sigma_{i=1}^{n}\beta_i$ upper paths or lower paths, then there must be m clause paths. These clause paths must use either an edge of the form $(v_{j,i,3}, v_{j',i,5})$ or an edge of the form $(v_{j,i,4}, v_{j',i,5})$. Moreover, by the construction of the graph H (by the edge of the form $(c_{j,3}, v_{j,i,b})$), each clause path corresponds to a Boolean variable in positive form (with the edge $(c_{j,3}, v_{j,i,3})$) or negative form (with the edge $(c_{j,3}, v_{j,i,3})$). This implies each clause path corresponds to the satisfaction of a clause in B according to the assignment A. Since there are m clause paths, all clauses in B are satisfied by the assignment A. This implies the Boolean expression B is satisfiable. □

Proof. Since the DAED4 problem is both in NP and NP-hard, then the DAED4 problem must be NP-complete. □

4.5.4.1 *Example*

As a visualization of H, consider an example of this graph construction when $B = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$, shown in Figure 5.2. This Boolean

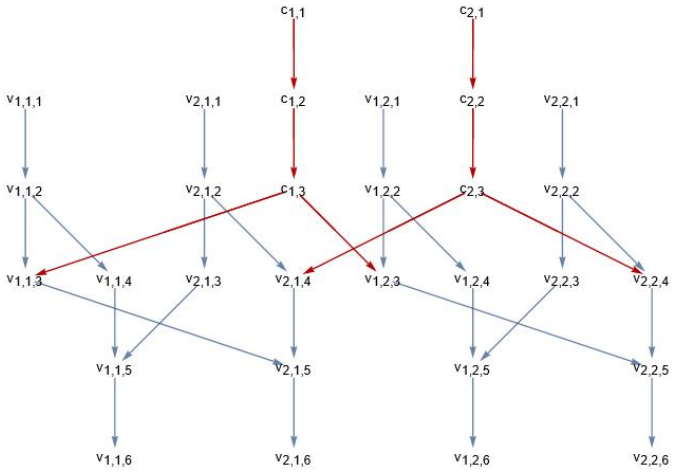


Figure 4.15: Graph representing the graph H of the Boolean expression $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

expression is True in the case where x_1 is True and x_2 is False, which is seen in Figure 4.16 using 6 4-paths: 2 (o_1) lower paths for the number of occurrences of x_1, 2 (o_2) upper paths for the number of occurrences of x_2, and 2 (m) clause paths for the number of clauses in the Boolean expression B.

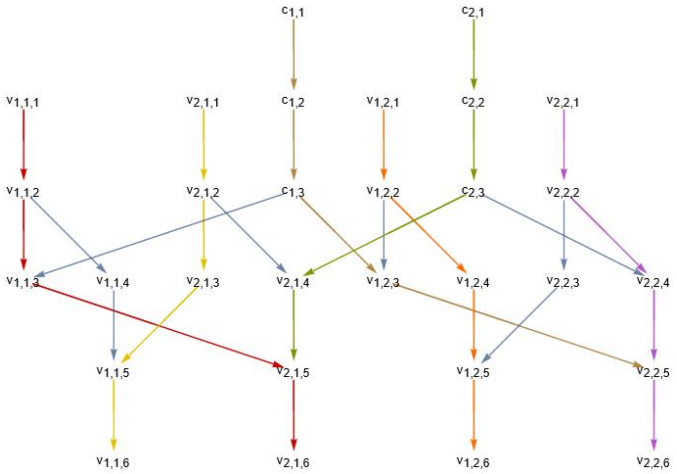


Figure 4.16: Possible 4-paths of the graph H that express x_1 as True and x_2 as False to satisfy the Boolean expression $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

4.5.5 *Extensions*

Corollary 4.5.4.1. *The DAEDk problem for $k > 3$ in the case of an unweighted directed acyclic graphs is NP-complete.*

Proof. In the preceding graph construction, consider the edges of the form $(v_{j,i,1}, v_{j,i,2})$ and the edges of the form $(c_{j,1}, c_{j,2})$. Each such edge may be replaced by a path graph of length $q \geq 1$ to prove without loss

of generality the NP-completeness of the DAED$(3+q)$ problem. Thus, using Theorem 4.5.3, this implies the DAEDk problem for $k > 3$ in the case of an unweighted directed acyclic graphs is NP-complete. □

Corollary 4.5.4.2. *The k-IP problem for $k > 3$ in the case of an unweighted directed acyclic graphs is NP-complete.*

Proof. Using Theorem 4.5.1 and the Corollary 4.5.4.1, this implies the k-IP problem for $k > 3$ in the case of an unweighted directed acyclic graphs is NP-complete. □

Note that unweighted directed acyclic graphs are a subclass of unweighted directed graphs, directed acyclic graphs and directed graphs. Therefore, Corollary 4.5.4.2 implies the NP-completeness of the k-IP for $k > 3$ for unweighted directed graphs, directed acyclic graphs and directed graphs.

4.6 APPROXIMATION ALGORITHMS

4.6.1 *Naïve approximation*

We consider now the k-IP problem over a simple unweighted directed acyclic graph $G = (V, E)$.

Theorem 4.6.1. *Any solution S to the k-IP problem that cannot be improved by adding a path is at least a $1/k$-approximation.*

Proof. Consider a set of disjoint paths $S = \{s_1, s_2, ..., s_h\}$ on G as a partial solution to the k-IP, such that the size of S cannot be improved by adding any k-path.

Consider a set of paths S_{OPT} in an optimal solution for the k-IP problem on G. Since S intersects every k-path of S_{OPT} on at least one edge, S covers at least $\lceil |S_{OPT}| \rceil$ edges. Since the graph G is unweighted, each k-path has the same interestingness score. This means any solution S is at least a $1/k$-approximation. □

We present the Algorithm 4, which outputs a set of paths with an interestingness score of at least a score of $1/k$ of the maximum interestingness score.

Theorem 4.6.2. *Algorithm 4 computes an $1/k$-approximation. It runs in $O(m \cdot u)$ time and $O(m)$ space, where u is the size of the output.*

Proof. This algorithm runs a topological sorting of G, and for each vertex assigns an LPath value in $O(m)$ time. Then, the maximum LPath is taken in $O(n)$ time for each loop. At least 1 path is added to the partial solution in $O(n)$ time and the LPath is updated in $O(m)$ time. This loop is repeated at most $O(u)$ times. This results in a total time complexity of $O(m \cdot u)$, and a total space complexity of $O(m)$, since the edge usage is tracked on G.

Algorithm 4 Naïve Solution

Require: Unweighted directed acyclic graph $G = (V, E)$

$S \leftarrow \{\}$
Longest path measure, defined in Section 1.8.3 (LPath) $\leftarrow$ LPath dictionary on G as seen in Section 1.8.3
$\alpha \leftarrow 0$
for v in V **do**
    if LPath$(v) > \alpha$ **then**
        $\alpha \leftarrow$ LPath(v)
        $\alpha_v \leftarrow v$
    end if
end for
while $\alpha \geq k$ **do**
    Find the α-path starting at v_i
    Add the corresponding $\lfloor LPath(i)/k \rfloor$ k-paths to S
    Mark all edges in the latter paths as used
    Update all values of the LPath dictionary, considering only unused edges
    for v in V **do**
        if LPath$(v) > \alpha$ **then**
            $\alpha \leftarrow$ LPath(v)
            $\alpha_v \leftarrow v$
        end if
    end for
end while
return S

Note, this algorithm's *while* loop condition implies the remaining graph has no k-path, since otherwise it would have been added to S. This guarantees the solution cannot be improved by adding a path, which proves this is an $1/k$-approximation by Theorem 4.6.1. □

4.6.2 *Linear Programming Approximation*

4.6.2.1 *Formulation*

Let $P = \{p_1, p_2, ..., p_{|P|}\}$ be the set of all possible k-paths in an unweighted directed graph $G = (V, E)$. Denote the set of values corresponding to each k-path as $x = \{x_1, x_2, ..., x_{|P|}\}$ where each $x_i \in \{0, 1\}$. For each edge $e_i \in E$, let $\Phi_i \subseteq P$ be the set of k-paths that contain the edge e_i.

Consider this reformulation of the k-IP problem as an integer linear programming problem (ILP) formulation:

$$\begin{aligned}&\text{Maximize } \mathbf{1}^T x\\ &\text{Subject to:}\\ &\qquad (\Sigma_{p_j \in \Phi_1} x_j, \Sigma_{p_j \in \Phi_2} x_j, \cdots, \Sigma_{p_j \in \Phi_m} x_j) \leq \mathbf{1}\\ &\qquad x \in \{0, 1\}^{|P|}\end{aligned}$$

where $\mathbf{1} = (1, 1, ..., 1)$ is a $|P|$-tuple of ones.

These constraints can be relaxed by allowing any $x_i \in x$ to a real number between 0 and 1. Here is the relaxed integer linear programming (RILP) formulation:

$$\begin{aligned}&\text{Maximize } \mathbf{1}^T x\\ &\text{Subject to:}\\ &\qquad (\Sigma_{p_j \in \Phi_1} x_j, \Sigma_{p_j \in \Phi_2} x_j, \cdots, \Sigma_{p_j \in \Phi_m} x_j) \leq \mathbf{1}\\ &\qquad \mathbf{0} \leq x \leq \mathbf{1}\end{aligned}$$

where $\mathbf{1}$ is a $|P|$-tuple of ones and $\mathbf{0}$ is a $|P|$-tuple of zeros.

Note, this approximation is motivated by the edge-disjoint path decomposition found in [8].

4.6.2.2 *Algorithm*

Consider the optimal solution OPT_{ILP} and OPT_{RILP} to the ILP and RILP formulations, respectively. Since the RILP formulation is less constrained than the ILP formulation, $\mathbf{1}^T\text{OPT}_{ILP} \leq \mathbf{1}^T\text{OPT}_{RILP}$. To minimize the integrality gap greedily, Algorithm 5 rounds the highest coefficient $x_h < 1$ in $\mathbf{x}$ to 1, adds the path p_h to a solution S and repeats the process until no k-path is left.

Algorithm 5 Linear Programming Approximation (Graph G)

Require: Unweighted directed graph $G = (V, E)$
1: $S \leftarrow \{\}$
2: Generate the set P of all k-paths based on G
3: Generate the constraints $\mathbf{c}$ based on P
4: **while** P is not empty **do**
5: $\mathbf{x} \leftarrow$ Solve the $RILP$ formulation based on $\mathbf{c}$
6: Add to S all paths $\in P$ of coefficient $x_i = 1$
7: Add to S the path $\in P$ corresponding to the highest $x_i \in \mathbf{x}$ that is not equal to 1
8: Remove from P all paths intersecting with S
9: Generate the constraints $\mathbf{c}$ based on P
10: **end while**
11: **return** S

While this algorithm experimentally provides a much better solution than the naïve approach, the approximation bound remains the same: $1/k$. This is because the algorithm stops when all paths have joint edges with the solution. The rounding scheme of the relaxed integer programming formulation does not provide a better approximation, since the rounding factor used may be arbitrarily low. As for the time complexity, the dual-method using [10] takes $O(m^{2.5})$ time for each linear programming change; thus, this has the total time complexity of $O(um^{2.5})$, where u is the size of the output.

4.6.3 *Colour Coding Approximation*

The current best algorithmic bound for approximating the 3-set packing problem, a variant of the packing problem, one of Karp's 21 NP-complete problems [45], is found in [67]. However, the best algorithmic bound for approximating the more general k-set packing problem was given by Cygan [12] and later improved by Fürer and Yu [23]. Both algorithms give an $\frac{1}{\frac{k+1}{3}+\epsilon}$-approximation for the k-IP problem.

Cygan's version has a lower time complexity for the datasets of interest, but the reader should note Fürer and Yu's version of the colour coding algorithm can also be applied and may have a better time complexity for large values of k. The algorithm uses as input f_0, that is, some initial solution. We consider f_0 to be the output of the "Naïve Solution" algorithm. It also uses the set of all k-paths in G, denoted $P = \{p_1, p_2, ...\}$. This method is outlined in Algorithm 6.

This algorithm can be summarized as a local search for an improving set of paths of size $r \leq 4(k+1)^{1/\epsilon}$, improving the solution until the solution cannot be improved after e^{c-1+ck} attempts. Each attempt

consists of the determination of a random set of paths based on colour coding techniques:

1. Randomly colour each edge by an element of $\{1, 2, \ldots, rk\}$. (line 8)
2. Randomly colour each k-path by an element of $\{rk+1, rk+2, \ldots, rk+r-1\}$. (line 9)
3. Randomly select a maximum set of k-paths X such that no colour is repeated in the set. (lines 18-26)
4. If the union of the initial solution f_0 and X forms a greater set than f_0, the attempt is successful. If not, the attempts are unsuccessful. (lines 28-32)

Theorem 4.6.3. *The solution provided by the algorithm 6 provides an* $\frac{1}{\frac{k+1}{3}+\epsilon}$*-approximation in polynomial time.*

The proof of this algorithm's approximation guarantee is found in [12]. Note, $\mathscr{DU}$ refers to the discrete uniform probability distribution, *ec* refers to edge colouring and *pc* refers to path colouring. However, we analyze here the time complexity of this algorithm. At worst, this algorithm uses the $u - u/k$ improvements from f_0. For each improvement, at most $4(k+1)^{1/\epsilon}$ values of r are considered. For each of those values, at most e^{c-1+ck} randomized trials are done, to maintain the theoretical guarantee. Considering the randomized colouring and the reduction of the set p, the number of edges, the number of sets and the number of edges per path, each trial has a complexity of $m^{k+1}k$. Overall, the time complexity of this algorithm is $O(4(k+1)^{1/\epsilon}m^{k+1}ku)$ where m is the number of edges and u is the size of the output.

4.7 EXACT ALGORITHMS

4.7.1 *General k-IP*

While the k-IP problem for $k \geq 4$ is NP-complete, practitioners may still consider exact algorithms of low exponential complexity to have the best interestingness scores. This is also motivated by the idea the output graph of the Mapper algorithm has in general under 1000 vertices [76].

For general weighted directed graphs, the k-IP problem can be formulated as a set cover problem. Consider a directed graph G, and the set of all possible k-paths in G. Consider each k-path to be a set of k elements. The goal is to find the maximum set cover, one of Karp's original NP-complete problems [45].

This formulation allows many algorithms intended to exactly solve the set cover problem to solve the general k-IP problem. Additionally,

Algorithm 6 Colour Coding Approximation

Require: Graph $G = (V, E)$, k, initial solution f_0, k-paths P, approximation parameter ϵ

$f_1 \leftarrow f_0$
while f_0 is equal to f_1 **do**
  $c \leftarrow 4(k+1)^{1/\epsilon}$
  for $r \leftarrow 2, r \leq c, r \leftarrow r+1$ **do**
    $c_0 \leftarrow 1$
    $t_{f_0} \leftarrow$True
    while $c_0 \leq e^{c-1+ck}$ and t_{f_0} **do**
      $ec \leftarrow \{ec_1, ec_2, ..., ec_m\} | \forall i(ec_i = \mathscr{DU}(1, rk))$
      $pc \leftarrow \{pc_1, pc_2, ..., pc_{|P|}\} | \ \forall i(pc_i = \mathscr{DU}(rk+1, rk+r-1))$
      $X \leftarrow \{\}$
      $xc \leftarrow \{\}$
      for each $p_i \in P$ **do**
        $l \leftarrow$ the list of colours corresponding to the edges of p_i
        if there is any duplicate colour in l **then**
          Remove p_i from P and pc_i from pc
        end if
      end for
      while P is not empty **do**
        Select randomly a path p_j from P
        while $pc_j \in xc$ **do**
          Remove p_j from P and pc_j from pc
          Select randomly a path p_j from P
        end while
        Add p_j to X and pc_j to xc
        Remove all paths in P that have the colours of the edges of p_j or the colour of p_j
      end while
      $f_{new} \leftarrow$ all paths of f_0 that do not intersect with X
      if $|f_0| < |f_{new}| + |X|$ **then**
        $t_{f_0} \leftarrow$False
        $f_0 \leftarrow f_{new} \cup X|$
        $f_1 \leftarrow f_0$
      end if
    end while
    $c_0 \leftarrow c_0 + 1$
  end for
end while
return f_0

in the case of weight-balanced multilayered[2] graphs, the sum of the weights of each k-path is at most $l \cdot k$, where l is the length of intervals. Thus, each set is constrained to a region of fixed size. This allows for the use of geometrical exact algorithms [11].

4.7.2 *Graphs of diameter k*

In the case of multilayered acyclic graphs, multiple cases may be easier to solve. The simplest case is if the graph diameter of a multilayered graph G is k. In this case, the k-IP problem is in P.

Theorem 4.7.1. *Assume the graph diameter of the multilayered acyclic graph G is k. Then, the k-IP problem on G is in P. This is shown in Algorithm 7.*

Proof. Consider a directed acyclic graph, where the graph diameter is k. Calculating the LPath takes $O(m+n)$ time. The "while" loop

Algorithm 7 Diameter k

Require: Multilayered Acyclic Graph $G = (V, E)$ of diameter k

```
Mark all edges in the path as unused
Calculate the LPath dictionary
α ← 0
Set S to an empty list
for v in V do
    if LPath(v) > α then
        α ← LPath(v)
        α_v ← v
    end if
end for
while α ≥ k do
    Find a k-path starting at α_v
    Add the corresponding k-path to S
    Mark all edges in the path as used
    Update all values of LPath considering only unused edges
    for v in V do
        if LPath(v) > α then
            α ← LPath(v)
            α_v ← v
        end if
    end for
end while
return S
```

searches for the α in $O(n)$ time, and explores $O(m)$ edges in $O(m)$ time. Considering the loop is repeated at most u times, where u is the size of the output, this algorithm has a $O((m+n)u)$ time complexity.

2 See Section 1.8.4 for multilayered graphs.

Thus, in the case of a graph diameter of k, the k-IP solution can be found in $O((m+n)u)$ time. □

Note, in the case of multilayered acyclic graphs, a maximum flow algorithm may be used instead [53].

1. Assign to every edge of G a flow capacity of 1.
2. Add a source vertex at level -1, and fully connect it to all vertices in level 0 with an infinite flow capacity.
3. Likewise, add a sink vertex at level k+1, and fully connect it to all vertices of level k with infinite flow capacity.
4. Using an efficient maximum flow algorithm in $O(nm)$ time [53], the optimal k-IP solution can be found.

The optimality of this method can be argued as follows. Consider f to be the size of the maximum flow and let u to be the maximum number of k-paths possible in G. Since the graph diameter is k, all k-paths in G must start at the layer 0 and end at layer k, thus, the maximum flow corresponds to the maximum number of k-paths. If there are more k-paths than maximum flow, a greater maximum flow may be determined using all the edges covered by the k-paths. This implies $f \geq u$. Since from the maximum flow u k-paths may be determined, $f \leq u$ is also true. Overall, in the case of multilayered acyclic graphs of diameter k, $f = u$ and the k-IP problem is in P.

4.7.3 *Fixed parameter tractable (FPT)*

One major source of complexity for the k-IP problem is the size of k. If k is $O(1)$, then the enumerations of all possible k-paths is possible, but the output u is practically expected to be $\Omega(n)$. If k is $\Omega(n)$, then the number of possible paths is exponential. Thus, the priority is given to assumptions ignoring k, such as restricting the the output u to $O(1)$ size.

This assumption is sufficient to motivate an FPT algorithm based on the output size u. We consider now the k-IP problem over an unweighted multilayered directed graph.

Theorem 4.7.2. *Consider the case where u and k are constant or u and $n-k$ are constant. Then, the k-IP problem is in P. This is shown in Algorithm 8.*

As defined in [49], the graph $G_k = (V_k, E_k)$ is constructed in the following manner, where its vertices are u-tuples of the vertices in G and edges are given between vertices of G_k where only one vertex of the tuple is changed from v_i to v_j and the difference in LPath ($LPath(v_i) - LPath(v_j)$) is non-negative :

$$V_k = \{(v_1, v_2, ..., v_u) \mid v_i \in V\}$$

Algorithm 8 Li, Mccormick, and Simchi-Levi Adaptation to k-IP

Require: Unweighted Graph $G = (V, E), |E| = m$

Form the graph $G_k = (V_k, E_k)$ mentioned in [49] from G, for the arc-disjoint case.
output $\leftarrow \{\}$
$u \leftarrow \frac{|E|}{2k}$
$u_{best} \leftarrow 0$
for each u such that $0 \leq u \leq \frac{|E|}{k}$ attempted in a binary search fashion **do**
    $t_u \leftarrow$ False ▷ A solution was not reached
    for each vertex $S_i = (s_1, s_2, ..., s_u) \in V_k$ **do**
        for each vertex $T_i = (t_1, t_2, ..., t_u) \in V_k$ such that $(\forall j)(layer(s_j) + k = layer(t_j))$ **do**
            Using Breadth-First search, find a path p from the vertex S_i to the vertex T_i in G_k.
            if p exists **then** ▷ A solution was reached
                if $u > u_{best}$ **then** ▷ It was the best solution so far
                    $u_{best} \leftarrow u$
                    $p_{best} \leftarrow p$
                end if
                Break from the two for loops
            end if
        end for
    end for
end for
return p_{best}

$$
\begin{aligned}
E_k = V_k = \{&(v_1, ..., v_{i-1}, v_i, v_{i+1}, ..., v_u) \to (v_1, ..., v_{i-1}, v_j, v_{i+1}, ..., v_u) \mid \\
&(v_1, ..., v_{i-1}, v_i, v_{i+1}, ..., v_u) \in V_k, \\
&(v_1, ..., v_{i-1}, v_j, v_{i+1}, ..., v_u) \in V_k, \\
&v_i \neq v_j, \\
&LPath(v_i) \geq LPath(v_j)\}
\end{aligned}
$$

Proof. For correctness, the proof of the graph construction G_k can be found in [49]. The k layers between the start vertex s_i and end vertex t_i ensure only k-paths are included in p. Indeed, by the definition of multilayered graphs in Section 1.8.4, for a vertex a of layer A and a vertex b of layer $A + k$, all paths from a to b have a length of k. For the space complexity, the graph G_k is of size $|E|^{u-1}$ (considering the reformulation of the problem from edge-disjoint to vertex-disjoint). For the time complexity, the algorithm searches through integer values of u going higher when t_u is true and lower when t_u is false. This leads to the highest value of u, thus u_{best}. That binary search has a $O(\log_2(\frac{|E|}{k}))$ time complexity, as indicated by its range. For multilayered graphs, we define the maximum number of vertices on any layer of G as w,[3] where $w << n$. Then, choosing the starting vertices has a complexity of $O(|V|^u)$, and considering these choices, choosing the terminal vertices has a complexity of $O(w^u)$. Finally, finding the path in G_k takes $O(|E|^{u-1}(|E||V|)) = O(|E|^u|V|)$ time. Since these operations enclose each other, their complexities multiply. Thus, the total time complexity of this algorithm is $O(|E|^u|V|^{u+1}w^u \log_2(\frac{|E|}{k}))$. □

4.8 SIMULATION

In this section, we tested our implementation of the naïve, the linear programming and the colour coding approximation algorithms. This implementation was done using Wolfram Mathematica [35]. The source code of these implementation and corresponding tests are found at https://github.com/MarkCycVic/EfficientInterestingPaths.

We use the output graphs of the Mapper algorithm found in the Kepler Mapper implementation [69]:

1. The cat dataset refers to a point-cloud sampled from a 3D model of a cat. The corresponding graph Mapper output has 20 vertices and 19 edges.

2. The breast cancer dataset refers to the Breast Cancer Wisconsin (Diagnostic) dataset [71], and reflects the use of topological data mining for medical science and biology. The corresponding graph Mapper output has 158 vertices and 304 edges.

3. The Modified National Institute of Standards and Technology (MNIST) dataset is a large database of handwritten digits that

3 This is referred to as graph width in [49].

is commonly used as a standard for machine learning [13], and reflects the use of topological data mining for machine learning. The corresponding graph Mapper output has 160 vertices and 369 edges.

Note these datasets are the standard datasets for the application of the Mapper algorithm [4, 34, 48, 62, 64, 68, 70, 72, 73]. We also tested the approximation algorithms on random graphs of at most 50 edges and relevant datasets [62, 70]. Note all output graphs are done with a tolerance of zero, meaning all graphs analyzed experimentally are unweighted directed acyclic graphs, since some approximation algorithms require those restrictions.

4.8.1 *Cat Dataset*

On the cat dataset for the 3-IP problem, the naïve approximation yields an interestingness score of 12.7122, the linear programming approximation yields an interestingness score of 19.0683 and the colour coding approximation also yields an interestingness score of 19.0683, where the optimal value is 19.0683. The 3-paths found are visually shown in Figure 4.17.

4.8.2 *MNIST Dataset*

On the MNIST dataset for the 4-IP problem, the naïve approximation yields an interestingness score of 177.137, the linear programming approximation yields an interestingness score of 201.075 and the colour coding approximation also yields an interestingness score of 191.500, where the optimal value is 201.075. The 4-paths found are visually shown in Figure 4.18.

For easier visualization, we consider the largest weakly connected component of the output graph of the MNIST dataset. This component has 23 vertices and 45 edges, and is situated in the upper-left corner of any graph in Figure 4.18. On this partial MNIST dataset for the 3-IP problem, the naïve approximation yields an interestingness score of 22.2464, the linear programming approximation yields an interestingness score of 28.6025 and the colour coding approximation also yields an interestingness score of 28.6025, where the optimal value is 28.6025. The 3-paths found are visually shown in Figure 4.19.

4.8.3 *Cancer Dataset*

On the cancer dataset for the 4-IP problem, the naïve approximation yields an interestingness score of 225.012, the linear programming approximation yields an interestingness score of 296.824 and the colour coding approximation also yields an interestingness score of

248.95, where the optimal value is unknown. The 4-paths found are visually shown in Figure 4.20.

4.8.4 *Random Graphs*

The following shows the performance of all approximation algorithms for the k-IP problem, attempted on various small graphs of between 5 and 50 vertices and between 5 and 50 edges. The edges generated were taken uniformly randomly from the set of all possible edges using the given vertices. The value of k varies between 3 and 6. The experimental results are shown in Table 4.1, where the score is the interestingness score measured divided by the optimal interestingness score.

Approximation Algorithm	Minimum Score	Average Score	Maximum Score
Naïve	0.2500	0.6830	1
Linear programming	0.5000	0.9891	1
Colour coding	0.7142	0.8919	1

Table 4.1: Approximation performance of all approximation algorithms on random small graphs

4.8.5 *Overview*

As for the dataset performance, we show in Table 4.2 the experimental interestingness score divided by the optimal score for that dataset.

Approximation Algorithm	Cat	MNIST	Cancer
Naïve	0.667	0.881	0.746
Linear programming	1	1	0.984
Colour coding	1	0.952	0.825

Table 4.2: Approximation performance of approximation algorithms for all datasets tested

4.9 DISCUSSION OF k-IP ALGORITHMS

Since the Mapper algorithm is intentionally an approximation of the dataset, it is coherent to propose approximation algorithms to its analysis.

Among our three approximation algorithms, the algorithm of lowest time complexity is the naïve approximation, followed by the linear programming approximation. However, these algorithms also have the worst approximation bound, $1/k$. For sparse random graphs where

the number of edges is $O(n)$, the randomized naïve approach is at worst an $1/4$-approximation. All algorithms perform on average worst on dense graphs, that is, when the number of edges is $\Omega(n^2)$. Since Mapper outputs are generally sparse [42], the Table 4.1 only considers sparse random graphs. Considering the linear programming approach has the best experimental results, both on random graphs and the datasets considered, it is more likely to be a preferred option for practitioners.

The colour coding algorithm offers the best approximation bound of all algorithms considered. However, it also implies a larger time complexity due to the required number of trials to have constant probability of the approximation bound, which makes it prohibitively expensive to run in practice. However, this algorithm may be more appropriate as a post-processing bound-guaranteeing algorithm for other approximation algorithms. Since a single trial is overall fast, a user-defined number of trials could be used to empirically confirm a given solution is close to the solution.

A major difficulty in calculating all paths in a given dataset is the size of k. For dense graphs, a linear increase in k may imply an exponential increase in compute time, making it difficult to evaluate the k-IP for large graphs when $k \geq 5$.

For exact algorithms, an extended version of the maximum flow approach may be used for values of k where $k > d/2$. However, for any k lower than d, the time complexity of this approach currently remains exponential. On the other hand, the fixed output tractable algorithm is applicable to a broader set of contexts, and may be considered more often on sparse graphs, where k is large and the expected size of the output is low.

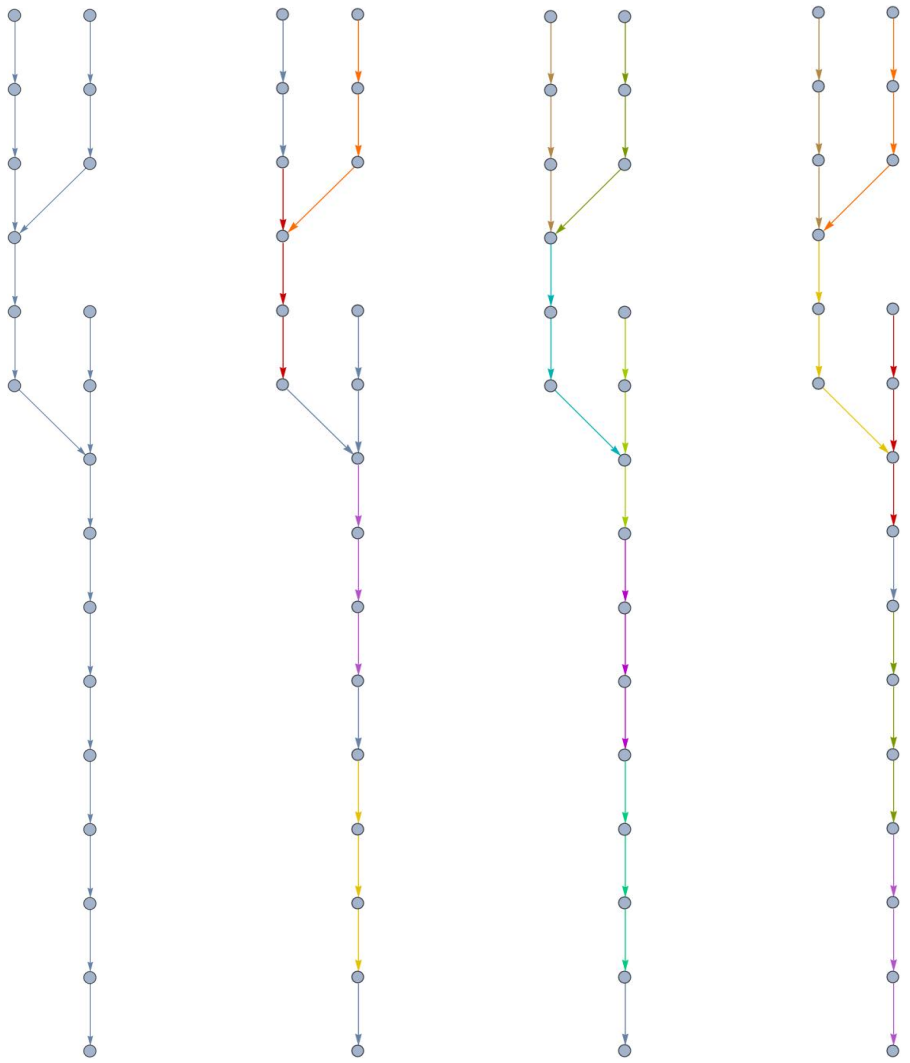


Figure 4.17: Solutions using 3-paths for the 3-IP problem on the cat dataset, where each colour represents a different path. From left to right: the initial output graph, the naïve approximation solution, the linear programming approximation solution and the colour coding approximation solution

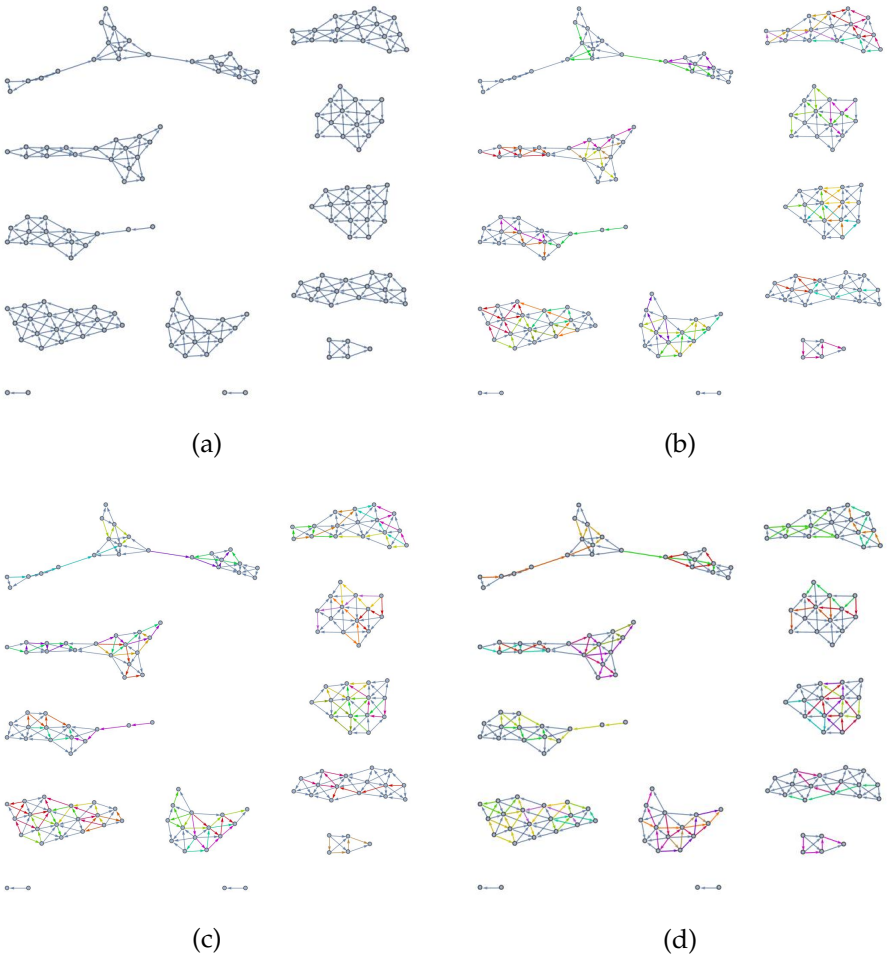
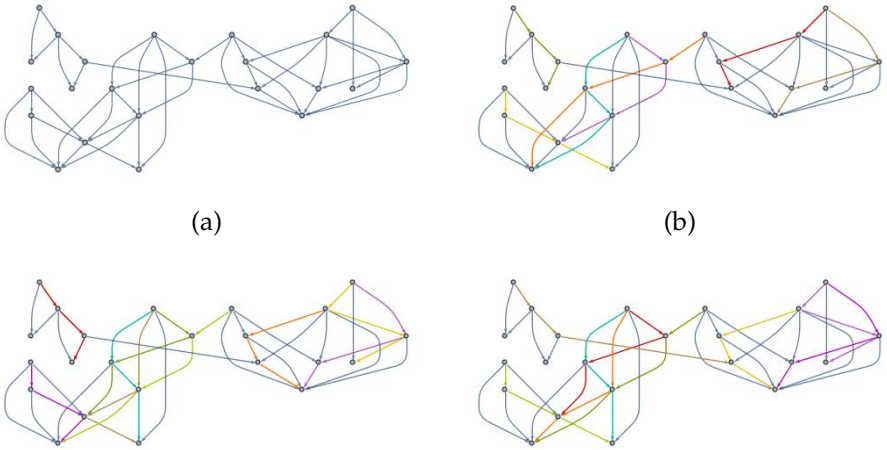


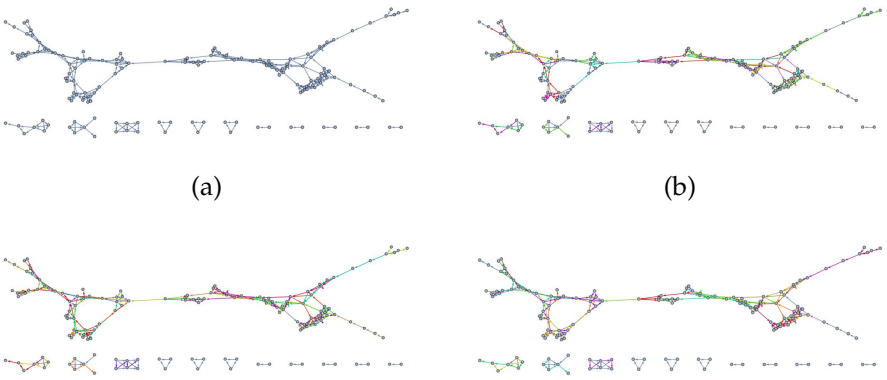
Figure 4.18: Solutions using 4-paths for the 4-IP problem on the largest weakly connected component of the MNIST dataset: (a) original output graph, (b) naive approximation, (c) linear programming approximation, (d) colour coding approximation.



(a) (b)

(c) (d)

Figure 4.19: Solutions using 3-paths for the 3-IP problem on the first weakly connected component of the MNIST dataset: (a) original output graph, (b) naive approximation, (c) linear programming approximation, (d) colour coding approximation.



(a) (b)

(c) (d)

Figure 4.20: Solutions using 4-paths for the 4-IP problem on the cancer dataset: (a) original output graph, (b) naive approximation, (c) linear programming approximation, (d) colour coding approximation.

5 THE IP PROBLEM

Similar to the k-IP problem, the IP problem maximizes the total interestingness score of multiple paths over a graph G. However, the paths of a solution to the IP problem may be of any positive length. Note, interestingness is defined in Section 1.9.

This section explores the IP problem, starting with its definition and motivation, followed by a proof of its NP-completeness. Then, we prove an approximation bound for a class of approximation algorithms for the IP problem. This motivates 3 heuristics based on this bound.

5.1 DEFINITION

Find the edge-disjoint set P of paths in $G = (V, E)$ such that the sum of their interestingness score is maximized (P will exactly cover E, i.e. each $e \in E$ is part of exactly one $p \in P$).

Consider also the decision form of this problem:

For a given weight t_0, decide whether there exists a collection P of edge-disjoint paths in G whose sum of interestingness scores is greater than or equal to t_0.

5.2 MOTIVATION

The main motivation of the IP problem is the unsupervised detection of flares. Generally, the goal of the Mapper algorithm is to summarize the features of complex datasets. The Max-IP enables the detection of a central path, but fails to detect concurrent flares and may break significant flares [42]. On the other hand, the k-IP formulation is akin to supervised machine learning in that it requires careful parametrization (k) relative to a measure (significant size of sample) is that practically difficult to determine. A high k may result in undetected flares, while a low k may result in noisy flares, but determining the correct k may be difficult in practice. Moreover, some flares may be bigger than other flares. Thus, the problem formulation should therefore reflect this reality. The IP in that sense is closer to an unsupervised method in the heuristic that the maximization of the total interestingness score should overall recognize the flares better than through parametrization.

5.3 NP-COMPLETENESS PROOF

Consider the 3D-matching problem, where $X = \{x_1, x_2, ..., x_n\}$, $Y = \{y_1, y_2, ..., y_n\}$ and $Z = \{z_1, z_2, ..., z_n\}$ are 3 pairwise disjoints sets of elements.

We write $x_i \leq x_j$ (respectively $y_i \leq y_j$ and $z_i \leq z_j$) whenever $i \leq j$, and we write $x_i < x_j$ (respectively $y_i < y_j$ and $z_i < z_j$) whenever $i < j$. Moreover, consider two triples (x, y, z) and (x', y', z'), where $x, x' \in X$, $y, y' \in Y$ and $z, z' \in Z$. We write $(x, y, z) < (x', y', z')$ whenever (i) $x < x'$, or (ii) $x = x'$ and $y < y'$, or (iii) $x = x'$, $y = y'$ and $z < z'$. This corresponds to the lexicographical order of the two triples.

We define vertex identification as follows: "The contraction of a pair of vertices v_i and v_j of a graph, also called vertex identification, is the operation that produces a graph in which the two nodes v_i and v_j are replaced with a single node v such that v is adjacent to the union of the nodes to which v_i and v_j were originally adjacent." [54]

We define graph union as follows: "The union $G = G_i \cup G_j$ of graphs G_i and G_j with disjoint point sets V_i and V_j respectively and edge sets E_i and E_j respectively is the graph G with $V = V_i \cup V_j$ and $E = E_i \cup E_j$." [28, 30]

Consider a set $\mathscr{C} \subseteq X \times Y \times Z$. The goal is to find the maximum size subset $\mathscr{C}' \subseteq \mathscr{C}$ such that every triple is pairwise disjoint. This problem is NP-complete when each element is contained in exactly 3 triples of $\mathscr{C}$, i.e., $\mathscr{C}'$ is a perfect matching in a 3-regular hypergraph corresponding to $\mathscr{C}$ [3, 44]. Note, this variation of the 3D matching problem implies the subset $\mathscr{C}'$ contains exactly once all elements of X, Y and Z.

5.3.1 *Certificate of polynomial length*

Lemma 5.3.1. *The IP problem is in NP.*

Proof. Consider a set P of paths in a weighted acyclic digraph $G = (V, E)$ where $|E| = m$ with a target weight of t_0. In $O(m^2)$ time, paths can be verified pairwise edge-disjoint. In $O(m)$ time, the total interestingness score of P can be evaluated and compared to t_0. Thus, the IP problem is in NP. □

5.3.2 *Graph construction*

In this section, we provide a construction of a directed acyclic graph $G = (V, E)$ from a given instance of the perfect matching in a 3-regular hypergraph $(X, \mathscr{C})$ problem, as mentioned in Section 5.3.

A weighted acyclic simple digraph G is constructed based on the graph union of isomorphic graphs. Consider any 3 different triples $T_1 = \{t_j, t_k, t_l\} \subseteq \mathscr{C}$ that

share the same $y_1 \in Y$ element, that is, of the form $T_1 = \{\{x, y_1, z\}, \{x', y_1, z'\}, \{x'', y_1, z''\} \mid x, x', x'' \in X, z, z', z'' \in Z\}$.

Without loss of generality, to simplify the notation moving forward, the elements of the triples of T_1 are denoted as follows: $T_1 = \{\{x, y_1, z\}, \{x', y_1, z'\}, \{x'', y_1, z''\}\}$. Moreover, we assume without loss of generality that $(x, y_1, z) < (x', y_1, z') < (x'', y_1, z'')$. Consider the graph Γ_1 representing T_1 shown in Figure 5.1. Note, since each triple contains one element of Y and each element of Y is contained in exactly 3 triples in $\mathscr{C}$, then each triple in $\mathscr{C}$ is represented by exactly one graph Γ_1.

We denote the vertices of Γ_1 as $v_{x,1}$, $v_{x,2}$,$v_{x',1}$, $v_{x',2}$,$v_{x'',1}$, $v_{x'',2}$, $v_{y_1,1}$, $v_{y_1,2}$, $v_{y_1,3}$, $v_{y_1,4}$, $v_{y_1,5}$, $v_{y_1,6}$, $v_{y_1,7}$, $v_{y_1,8}$, $v_{z,1}$, $v_{z,2}$,$v_{z',1}$, $v_{z',2}$,$v_{z'',1}$, $v_{z'',2}$. The form $v_{w,r}$ may be read as the r^{th} *vertex of* w, where w is an element of either X, Y or Z. This is shown in Figure 5.1. Each element of X has two vertices associated to it, each element of Y has eight vertices associated to it, and each element of Z has two vertices associated to it.

The edges in Γ_1 are the following: $e_x = (v_{x,1}, v_{x,2})$, $e_{x'} = (v_{x',1}, v_{x',2})$, $e_{x''} = (v_{x'',1}, v_{x'',2})$, $e_{y_1,1} = (v_{x,2}, v_{y_1,1})$, $e_{y_1,2} = (v_{y_1,1}, v_{y_1,2})$, $e_{y_1,3} = (v_{y_1,2}, v_{y_1,4})$, $e_{y_1,4} = (v_{x',2}, v_{y_1,3})$, $e_{y_1,5} = (v_{y_1,3}, v_{y_1,4})$, $e_{y_1,6} = (v_{x'',2}, v_{y_1,4})$, $e_{y_1,7} = (v_{y_1,4}, v_{y_1,5})$, $e_{y_1,8} = (v_{y_1,5}, v_{z,1})$, $e_{y_1,9} = (v_{y_1,5}, v_{y_1,6})$, $e_{y_1,10} = (v_{y_1,6}, v_{z',1})$, $e_{y_1,11} = (v_{y_1,5}, v_{y_1,7})$, $e_{y_1,12} = (v_{y_1,7}, v_{y_1,8})$, $e_{y_1,13} = (v_{y_1,8}, v_{z'',1})$, $e_z = (v_{z,1}, v_{z,2})$,$e_{z'} = (v_{z',1}, v_{z',2})$, $e_{z''} = (v_{z'',1}, v_{z'',2})$ and $e_z = (v_{z,1}, v_{z,2})$. We denote each edge representing an element w of X or Z as e_w and each edge representing an element w of Y as $e_{w,r}$, where r is the index of the corresponding edge as seen in Figure 5.1.

For each y_j, consider a graph Γ_j isomorphic to Γ_1. Note, since each triple contains one element of Y and each element of Y is contained in exactly 3 triples in $\mathscr{C}$, then each triple in $\mathscr{C}$ is represented by exactly one graph Γ_j.

For elements of X or Z, there are two vertices and one edge. There is one edge associated to each element of X or Z, denoted e_w, where w is the element. For each element of Y, there are 8 vertices and 13 edges.

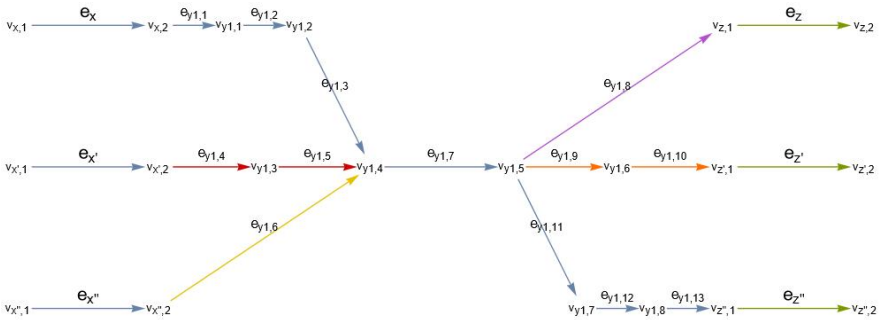


Figure 5.1: Graph Γ_1 representing the triples t_j, t_k, t_l for a given $y_1 \in Y$

Note, each edge has a weight, determined by trial-and-error and integer linear optimization given some constraints explained in Section 5.3.2.1.

1. Blue edges: 1
2. Red edges: $a = 363052$
3. Yellow edge: $b = 1127044$
4. Purple edge: $c = 364776$
5. Orange edge: $d = 105519$
6. Green edge: $w_z = 1463459$

As each triple in $\mathscr{C}$ is represented by exactly one graph Γ_j, the set of all Γ_j graphs may represent $\mathscr{C}$.

We define the graph $G = (V, E)$ as the graph union of all Γ_j graphs:

$$G = \cup_{j=1}^{n} \Gamma_j \tag{5.1}$$

The following is implied by the definition of graph union:

$$V = \cup_{j=1}^{n} (V_j \mid \Gamma_j = (V_j, E_j)) \tag{5.2}$$

$$E = \cup_{j=1}^{n} (E_j \mid \Gamma_j = (V_j, E_j)) \tag{5.3}$$

Overall, there are 13 edges for each element in Y, and one edge for each element in X or Y, thus $15n$ edges in total. Considering the union of $O(n)$ graphs of $O(1)$ edges, this graph construction is made in $O(n)$ time.

5.3.2.1 *Weight determination*

This proof is largely computational. The shape of the subgraph $\Gamma_1 = (V_1, E_1)$ was discovered by trial-and-error, to allow certain properties while minimizing the size of the graph and the weights. Consider the subgraph of Figure 5.1 and the weight $\alpha \approx 7752729.24$. The following can computationally be verified using the weighted edges mentioned previously, α and the interestingness score formula, that is $I(P) = \Sigma_{i=1}^{|P|} w_{e_i} \log_2(i+1)$. Note, this path cover of Γ_1 includes exactly one element of $\{e_x, e_{x'}, e_{x''}\}$ and one element of $\{e_z, e_{z'}, e_{z''}\}$.

1. The sum of interesting scores for the paths from $\{e_x, e_{y_1,1}, e_{y_1,2}, e_{y_1,3}, e_{y_1,7}, e_{y_1,8}, e_z\}$, $\{e_{y_1,4}, e_{y_1,5}\}$, $\{e_{y_1,6}\}$, $\{e_{y_1,9}, e_{y_1,10}\}$, $\{e_{y_1,11}, e_{y_1,12}, e_{y_1,13}\}$ is α.

2. The sum of interesting scores for the paths from $\{e_{x'}, e_{y_1,4}, e_{y_1,5}, e_{y_1,7}, e_{y_1,9}, e_{y_1,10}, e_{z'}\}$, $\{e_{y_1,1}, e_{y_1,2}, e_{y_1,3}\}$, $\{e_{y_1,6}\}$, $\{e_{y_1,8}\}$, $\{e_{y_1,11}, e_{y_1,12}, e_{y_1,13}\}$ is α.

3. The sum of interesting scores for the paths from $\{e_{x''}, e_{y_1,6}, e_{y_1,7}, e_{y_1,8}, e_{y_1,11}, e_{y_1,12}, e_{y_1,13}, e_{z''}\}$, $\{e_{y_1,1}, e_{y_1,2}, e_{y_1,3}\}$, $\{e_{y_1,4}, e_{y_1,5}\}$, $\{e_{y_1,8}\}$, $\{e_{y_1,9}, e_{y_1,10}\}$ is α.

4. The sum of interesting scores for the paths from $\{e_x, e_{y_1,1}, e_{y_1,2}, e_{y_1,3}, e_{y_1,7}, e_{y_1,9}, e_{y_1,10}, e_{z'}\}$, $\{e_{y_1,4}, e_{y_1,5}\}$, $\{e_{y_1,6}\}$, $\{e_{y_1,8}\}$, $\{e_{y_1,11}, e_{y_1,12}, e_{y_1,13}\}$ is less than or equal to $\alpha - 1$.

5. The sum of interesting scores for the paths from $\{e_x, e_{y_1,1}, e_{y_1,2}, e_{y_1,3}, e_{y_1,7}, e_{y_1,11}, e_{y_1,12}, e_{y_1,13}, e_{z''}\}$, $\{e_{y_1,4}, e_{y_1,5}\}$, $\{e_{y_1,6}\}$, $\{e_{y_1,8}\}$ and $\{e_{y_1,9}, e_{y_1,10}\}$ is less than $\alpha - 1$.

6. The sum of interesting scores for the paths from $\{e_{x'}, e_{y_1,4}, e_{y_1,5}, e_{y_1,7}, e_{y_1,8}, e_z\}$, $\{e_{y_1,1}, e_{y_1,2}, e_{y_1,3}\}$, $\{e_{y_1,6}\}$, $\{e_{y_1,9}, e_{y_1,10}\}$ and $\{e_{y_1,11}, e_{y_1,12}, e_{y_1,13}\}$ is less than or equal to $\alpha - 1$.

7. The sum of interesting scores for the paths from $\{e_{x'}, e_{y_1,4}, e_{y_1,5}, e_{y_1,7}, e_{y_1,11}, e_{y_1,12}, e_{y_1,13}, e_{z''}\}$, $\{e_{y_1,1}, e_{y_1,2}, e_{y_1,3}\}$, $\{e_{y_1,6}\}$, $\{e_{y_1,8}\}$ and $\{e_{y_1,9}, e_{y_1,10}\}$ is less than or equal to $\alpha - 1$.

8. The sum of interesting scores for the paths from $\{e_{x''}, e_{y_1,6}, e_{y_1,7}, e_{y_1,8}, e_z\}$, $\{e_{y_1,1}, e_{y_1,2}, e_{y_1,3}\}$, $\{e_{y_1,4}, e_{y_1,5}\}$, $\{e_{y_1,9}, e_{y_1,10}\}$ and $\{e_{y_1,11}, e_{y_1,12}, e_{y_1,13}\}$ is less than or equal to $\alpha - 1$.

9. The sum of interesting scores for the paths from $\{e_{x''}, e_{y_1,6}, e_{y_1,7}, e_{y_1,9}, e_{y_1,10}, e_{z'}\}$, $\{e_{y_1,1}, e_{y_1,2}, e_{y_1,3}\}$, $\{e_{y_1,4}, e_{y_1,5}\}$, $\{e_{y_1,8}\}$ and $\{e_{y_1,11}, e_{y_1,12}, e_{y_1,13}\}$ is less than or equal to $\alpha - 1$.

In other words, the path partition of the graph Γ_1 including exactly one edge among e_x, $e_{x'}$ and e_x, and one edge among e_z, $e_{z'}$ and $e_{z''}$ is maximum when it is equal to α, and the only way to get that total score is to use either a path for e_x to e_z, a path for $e_{x'}$ to $e_{z'}$ or a path for $e_{x''}$ to $e_{z''}$. If any other path uses the central edge $e_{y_1,7}$, then the total score is at most $\alpha - 1$.

Note, the weights chosen are based on the integer linear minimization of weights under the preceding constraints.

The interestingness score of choosing the path from x_p to z_q, denoted $I(p,q)$ is equal to

$$\begin{aligned} I(p,q) &= \log_2(2) + w_p \log_2(2 + l_p) + \log_2(3 + l_p) \\ &\quad + w_q \log_2\left(\frac{(3 + l_p + l_q)!}{(3 + l_p)!}\right) + w_z \log_2(4 + l_p + l_q) \\ &\quad + I(P_1) + I(P_2) + I(P_3) + I(P_4), \end{aligned} \tag{5.4}$$

where $l_p \in [1,3]$ refers to the number of y edges from p leading to $v_{y_1,4}$ with weight $w_p \in \{1,a,b\}$, $l_q \in [1,3]$ refers to the number of y edges starting from $v_{y_1,5}$ leading to q with weight $w_q \in \{1,c,d\}$ and P_1, P_2, P_3 and P_4 refer to the four smaller paths that remain after the choice of the path from p to q. From this definition of maximum subgraphs, all constraints may be formulated:

$$\begin{aligned} I(x,z) = I(x',z') = I(x'',z'') &= \alpha \\ \alpha &> I(x,z')+1 \\ \alpha &> I(x,z'')+1 \\ \alpha &> I(x',z)+1 \\ \alpha &> I(x',z'')+1 \\ \alpha &> I(x'',z)+1 \\ \alpha &> I(x'',z')+1 \\ a,b,c,d,w_z &> 0 \end{aligned} \tag{5.5}$$

Considering the linear minimization of the objective function $a+b+c+d+w_z$, the weights a,b,c,d,w_z,α were previously given.

5.3.2.2 *Example*

As a visualization of G, consider an example of this graph construction when $\mathscr{C}_e = \{\{x_1,y_1,z_1\}, \{x_2,y_1,z_2\}, \{x_3,y_1,z_3\}, \{x_1,y_2,z_1\},\{x_3,y_2,z_3\}, \{x_4,y_2,z_4\}\}$, shown in Figure 5.2. The red edges represent all triples with the element y_1 as the second element. The yellow edges represent all triples having the element y_2 as the second element. There are multiple possible graphs G_e as any triple may be considered the first, second, or third element on y. This has no bearing on the rest of this proof, as it refers to **any** graph G of $\mathscr{C}$.

5.3.3 *Reduction: from 3D matching to IP*

Lemma 5.3.2. *Consider a set $\mathscr{C}$ of triples given in the previously mentioned 3D matching problem. If $\mathscr{C}$ has a perfect matching $\mathscr{C}'$ on X, Y and Z, then the IP solution P of G has a total interestingness score of $t_0 = \alpha \cdot n$.*

Proof. Consider the triple $t_i = \{x_q, y_r, z_s\}$ chosen out of the 3 triples intersecting at y_r. For the graph associated to y_r, choose the corresponding path from e_{x_q} to e_{z_s} and the remaining smaller paths. Consider this to be a partial solution P_i. As noted in Section 5.3.2.1, this yields exactly α. Since each triple in $\mathscr{C}'$ is disjoint, each partial solution in G for each $y_1 \in Y$ is also disjoint. The union of each partial solution constitutes an exact path cover of G since the previously mentioned 3D matching problem is an exact set cover of X, Y and Z. This implies all e_x edges, e_y edges, and e_z edges are covered exactly once, corresponding to how

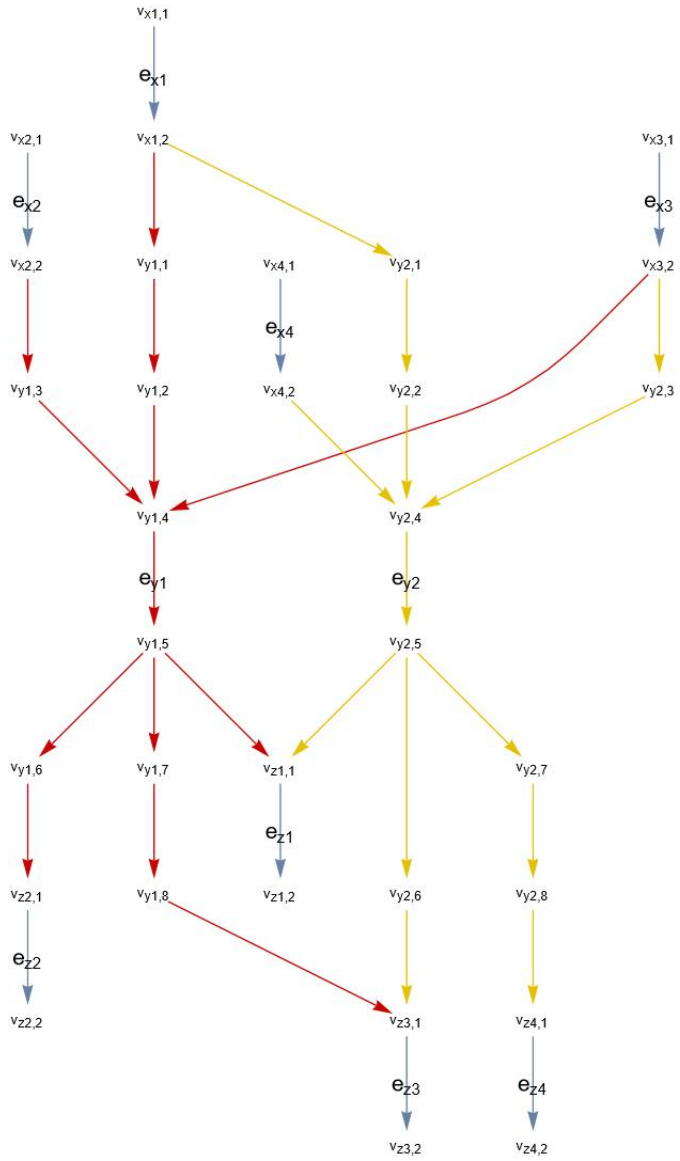


Figure 5.2: Graph representing G_e

each element $x \in X$, $y \in Y$ and $z \in Z$ is included exactly once in $\mathscr{C}'$. This implies the union of all partial solutions is a valid solution and has a total interestingness score of exactly $t_0 = \alpha \cdot n$. □

5.3.4 *Reduction: from IP to 3D matching*

Lemma 5.3.3. *If the IP solution P of G has a total interestingness score of $t_0 = \alpha \cdot n$, then $\mathscr{C}$ has a perfect matching $\mathscr{C}'$ on X, Y and Z.*

Proof. Consider the set of paths P which solves the IP problem on G_T. Since the total interestingness score is $t_0 = \alpha \cdot n$, it implies the average score per subgraph Γ_1 of $y_1 \in Y$ is α. However, α is the maximum interestingness score for any subgraphs Γ_1. This implies each subgraph

Γ_1 has an interestingness score of exactly α. The only way to obtain this score is that the main path chosen in each decomposition corresponds to a triple in $\mathscr{C}$. Note the edges $e_{x_q}, e_{y_r,7}$ and e_{z_i} ensure each main path is disjoint from another path in P. Consider the set of all paths containing e_{x_q} edges. This must correspond to all the main paths. Otherwise, some subgraph does not have a maximum interestingness score, which is a contradiction. For each path, consider the index q, r, s of the edges $e_{x_q}, e_{y_r,7}$ and e_{z_i} as the triple $\{t_q, t_r, t_s\}$. Form the solution $\mathscr{C}'$ from each triple. These triples correspond to some element of $\mathscr{C}$ as this is the only way the interestingness score of the corresponding subgraph is α. Also, all elements of X, Y and Z are disjoint: by contradiction, if some elements are not disjoint, this implies P is not edge-disjoint, which is a contradiction. □

Proof. Since the 3-IP problem is both in NP and NP-hard, then the 3-IP problem must be NP-complete. □

5.4 APPROXIMATION BOUND

Consider an unweighted directed acyclic graph $G = (V, E)$. The goal is to find an edge-disjoint path partition P of the total interestingness score

$$\Sigma_{j=1}^{|P|} I(P_j) = \Sigma_{j=1}^{|P|} \Sigma_{i=1}^{|P_j|} \log_2(i+1) \tag{5.6}$$

of paths in maximized.

Theorem 5.4.1. *The maximization of the sum of interestingness scores for each path can equivalently be reformulated as the maximization of the average interestingness score of every edge, calculated as* $\log(i_e + 1)$*, where* i_e *is the rank of* e *within its path in the solution* P*.*

Proof. Since the solution P is an exact path cover of G, it also implies P is an edge cover of G. Thus:

$$\begin{aligned} \max(\Sigma_{j=1}^{|P|} I(P_j)) &= \max(\Sigma_{j=1}^{|P|} \Sigma_{i=1}^{|P_j|} \log_2(i+1)) \\ &= \max(\Sigma_{e \in E} \log_2(k_e + 1)) \\ \frac{\max(\Sigma_{j=1}^{|P|} I(P_j))}{|E|} &= \frac{\max(\Sigma_{e \in E} \log_2(k_e + 1))}{|E|} \end{aligned}$$

Considering $|E|$ is a constant factor, this constitutes an equivalent reformulation. □

This theorem demonstrates that while it is important to maximize the number of large paths, greedily taking maximum paths may lead to multiple shorter paths.

Note, the edge-disjoint exact path cover problem is in P. Consider an arbitrary vertex v_i with a degree difference, defined as:

$$d_{v_i} = outdegree(v_i) - indegree(v_i) \tag{5.7}$$

Considering the degree sum formula, the total of all indegree and outdegree is equal. Consider all vertices with a positive d_{v_i}, this implies at least d_{v_i} paths start at v_i. Consider all vertices with a negative d_{v_i}, this implies at least d_{v_i} paths end at v_i.

Consider the following path removal technique over the directed acyclic graph G:

Algorithm 9 Generalized Greedy Minimum Paths

Require: Direct acyclic graph G
1: $P' \leftarrow \{\}$
2: **while** G is not empty **do**
3: Consider a vertex v where $d_v > 0$
4: Find a path p starting at v and ending at a vertex w such that $d_w < 0$.
5: Add p to P'
6: Remove all edges in p from G.
7: **end while**
8: **return** P'

Theorem 5.4.2. *The algorithm 9 generates an exact path cover of G.*

Proof. Consider an arbitrary directed acyclic graph G. By definition, if G has edges, the root node v_r of G has an indegree of 0 and a positive outdegree. Consider now forming a path starting at v_r. For each new vertex v encountered, there are two cases.

1. The degree difference is non-negative. Considering it has an indegree of at least one, it must have an outdegree of at least one as well. Therefore, the path may be extended to the next vertex.

2. The degree difference is negative. In this case, the condition is reached, the path may end and be removed from G.

The fact G is acyclic implies each vertex is visited at most once for each path. Considering the number of edges in G is finite, the second case is always reached. Considering the subgraph of a directed acyclic graph is also a directed acyclic graph, then the preceding statements are appliable to G and all its subgraphs. □

Theorem 5.4.3. *The algorithm 9 generates an exact path cover of G using a minimum number of paths.*

Proof. Consider the set S of all vertices of positive d_{v_i}. By its construction, the algorithm 9 generates exactly $\Sigma_{v_i \in S} d_{v_i}$ paths. By contradiction,

consider an exact path cover P of G for which $|P| < \Sigma_{v_i \in S} d_{v_i}$. This implies there exists a vertex v' in S for which less than $d_{v'}$ paths start. This implies at least one edge connected to v' is not covered by P, which is a contradiction of the fact P is an exact path cover. □

For simplicity, we define a *minimum-path solution* as any exact path partition of G with the minimum number of paths.

Consider the number of edges in G to be m, and the minimum number of paths to be u. As mentioned in the introduction of this section, the problem can be considered the maximization of the average weight of every edge, considering the weight of every edge is dependent on the length of its path.

The average weight of every edge is maximized in the case where all edges are part of the same path since every edge contributes to the increase in the logarithmic factor. Considering the minimum number of paths is u, the large path must be separated in u paths. Consider taking $u-1$ edges from the large path to create one path of length $m-u+1$ and $u-1$ paths of 1 edge. For all cases where $m > u, m > 0$ and $u > 0$, Equation 5.8 holds:

$$\log_2(m-u+2)! + u - 1 > \Sigma_{i=1}^{u} \log_2(k_i+1)! \mid \Sigma_{i=1}^{u} k_i = m \wedge \exists k_i > 1 \tag{5.8}$$

This is based on the inequality $\log_2(m!/u!) > \log_2(m-u)! \mid m > u$. Inversely, consider taking u paths of length m/u. For simplicity, we assume here m is divisible by u. For all cases where $m > u, m > 0 and u > 0$, the Equation 5.9 holds:

$$u \log_2(m/u+1)! < \Sigma_{i=1}^{u} \log_2(k_i+1)! \mid \Sigma_{i=1}^{u} k_i = m \wedge \exists k_i < m/u \tag{5.9}$$

Thus, in the worst case, all edges are equally distributed between all u paths, such that the average edge factor is as close to 1 ($\log_2 2$) as possible. Thus, the best case may be expressed as $\log_2((m-u+2)!) + u - 1$ and the worst case can be expressed as $p \log_2((m/u+1)!)$. This leads to the approximation factor of A in Equation 5.10.

$$A \geq \frac{u \log_2((m/u+1)!)}{\log_2((m-u+2)!) + u - 1} \tag{5.10}$$

However, possible simplifications of the Equation 5.10 do not lead to a conclusive strict bound on A. Thus, we consider a realistic assumption in the context of the Mapper algorithm: u is relatively much smaller than m. Numerical analysis reveals that if $u \leq \sqrt{m}$, the minimum value of the Equation 5.10 is approximately 0.45321191, which is a 9/20-approximation. This implies various strategies and algorithms are possible under that bound, as long as they are proven to be minimum-path solutions.

5.5 HEURISTIC 1: LONGEST PATHS

The first strategy is proposed in [41], and can be considered the greedy application of the Max-IP algorithm (seen in Section 3.4 from [41]) until the graph G is completely partitioned. The algorithm 10 applies this heuristic:

Algorithm 10 Longest Path IP Approximation

Require: Unweighted Acyclic Directed Graph G
1: $S \leftarrow \{\}$
2: **while** G is not empty **do**
3: Find the longest path p in G using the Max-IP algorithm
4: Add p to the list S
5: Remove all edges of p from G
6: **end while**
7: **return** S

Theorem 5.5.1. *The Longest Path IP Approximation algorithm generates an exact path partition of G with the minimum number of paths in $O(u \cdot m)$ time.*

Proof. For each path in S, the algorithm uses a maximum interestingness score path p. This implies the first vertex of p has an indegree of 0 and the last vertex has an outdegree of 0. Otherwise, by contradiction, if the first vertex of the path has a positive indegree or the last vertex has a positive outdegree, then the path can be extended by at least one edge, which is a contradiction to the fact it is a maximum interestingness score path. Without loss of generality, recursively, it implies the start vertex of p has an outdegree greater than its indegree, and the end vertex of p has an indegree greater than its outdegree. This implies the number of paths is equal to the sum of positive degree differences. In turn, this implies this algorithm provides a minimum-path solution.

For each path in S, the Max-IP algorithm is applied, which has a complexity of $O(m)$. This is applied u times. Thus, this algorithm has a time complexity that is $O(u \cdot m)$. □

5.6 HEURISTIC 2: CLOSE PATHS

The second strategy is to select greedily paths of lowest length to avoid breaking paths in later choices, recursively, until the graph G is completely partitioned. Algorithm 11 applies this heuristic:

Theorem 5.6.1. *The Close paths IP Approximation algorithm generates an exact path partition of G with the minimum number of paths in $O(um)$ time.*

Proof. For each path in S, the algorithm uses a path with start and end vertices having an indegree of 0 and an outdegree of 0 respectively.

Algorithm 11 Close paths IP Approximation

Require: Unweighted Acyclic Directed Graph G

$S \leftarrow \{\}$
$\alpha \leftarrow 0$
LPath $\leftarrow$ the longest path measure dictionary 1.8.3 in G
while G is not empty **do**
    for v in V **do**
        if LPath$(v) > \alpha$ **then**
            $\alpha_v \leftarrow v$
        end if
    end for
    Find the shortest path p in G from α_v to the closest vertex with $outdegree = 0$
    Add p to the list S
    Remove all edges of p from G
    Update the LPath dictionary
end while
return S

In the recursive cases, it implies the start vertex has an outdegree greater than its indegree, and the end edge has an indegree greater than its outdegree. This implies the number of paths is equal to the sum of positive degree differences. In turn, this implies this algorithm provides a minimum-path solution.

For each vertex, the LPath dictionary is generated in $O(m+n)$ time. Then, finding α_v takes $O(m)$ time and the shortest path is found in $O(m)$ time. This is applied u times. Thus, this algorithm has a time complexity that is $O(u \cdot m)$. □

5.7 HEURISTIC 3: FLOW PATHS

The third strategy can be considered the greedy application of flow graphs that strike a balance between smaller complete paths and long paths. Algorithm 12 applies this heuristic:

Theorem 5.7.1. *The Max Flow IP Approximation algorithm generates an exact path partition of G with the minimum number of paths in $O(u \cdot m)$ time.*

Proof. For each path in S, the algorithm a path with a first and last vertices have an indegree of 0 and an outdegree of 0 respectively. In the recursive cases, it implies the start vertex has an outdegree greater than its indegree, and the end edge has an indegree greater than its outdegree. This implies the number of paths is equal to the sum of positive degree differences. In turn, this implies this algorithm provides a minimum-path solution.

Algorithm 12 Max Flow IP Approximation

Require: Unweighted Acyclic Directed Graph G

$S \leftarrow \{\}$
Calculate the LPath dictionary
while G is not empty **do**
    Apply the maximum flow algorithm in G as seen in [53], where all vertices of maximum LPath are sources and all vertices of LPath of 0 are sinks, and set F to be the set of all flow paths
    Add each $p_i \in P$ to the list S
    Remove all edges of each $p_i \in P$ from G
end while
return S

The max flow paths are found using the algorithm for Max Flow [53] on G, for a total time complexity of $O(m \cdot n)$. This is applied u times. Thus, this algorithm has a time complexity that is $O(u \cdot m \cdot n)$. □

5.8 SIMULATION

In this section, we tested our implementation of the long, the close, and flow approximation algorithms. This implementation was done using Wolfram Mathematica [35]. The source code of these implementation and corresponding tests are found at https://github.com/MarkCycVic/EfficientInterestingPaths.

We use the output graphs of the Mapper algorithm found in the Kepler Mapper implementation [69]:

1. The cat dataset refers to a point-cloud sampled from a 3D model of a cat. The corresponding graph Mapper output has 20 vertices and 19 edges.

2. The breast cancer dataset refers to the Breast Cancer Wisconsin (Diagnostic) dataset [71], and reflects the use of topological data mining for medical science and biology. The corresponding graph Mapper output has 158 vertices and 304 edges.

3. The Modified National Institute of Standards and Technology (MNIST) dataset is a large database of handwritten digits that is commonly referred to as a standard for machine learning [13], and reflects the use of topological data mining for machine learning. The corresponding graph Mapper output has 160 vertices and 369 edges.

Note these datasets are the standard datasets for the application of the Mapper algorithm [4, 34, 48, 62, 64, 68, 70, 72, 73]. Note all output graphs are tested with a tolerance of zero, meaning all graphs analyzed experimentally are directed acyclic unweighted digraphs, since some approximation algorithms require those restrictions.

5.8.1 *Cat Dataset*

On the cat dataset for the IP, the long approximation yields an interestingness score of 32.8691, the close approximation yields an interestingness score of 29.2055 and the flow approximation also yields an interestingness score of 32.8691, where the optimal value is 32.8691. The paths found are visually shown in Figure 5.3.

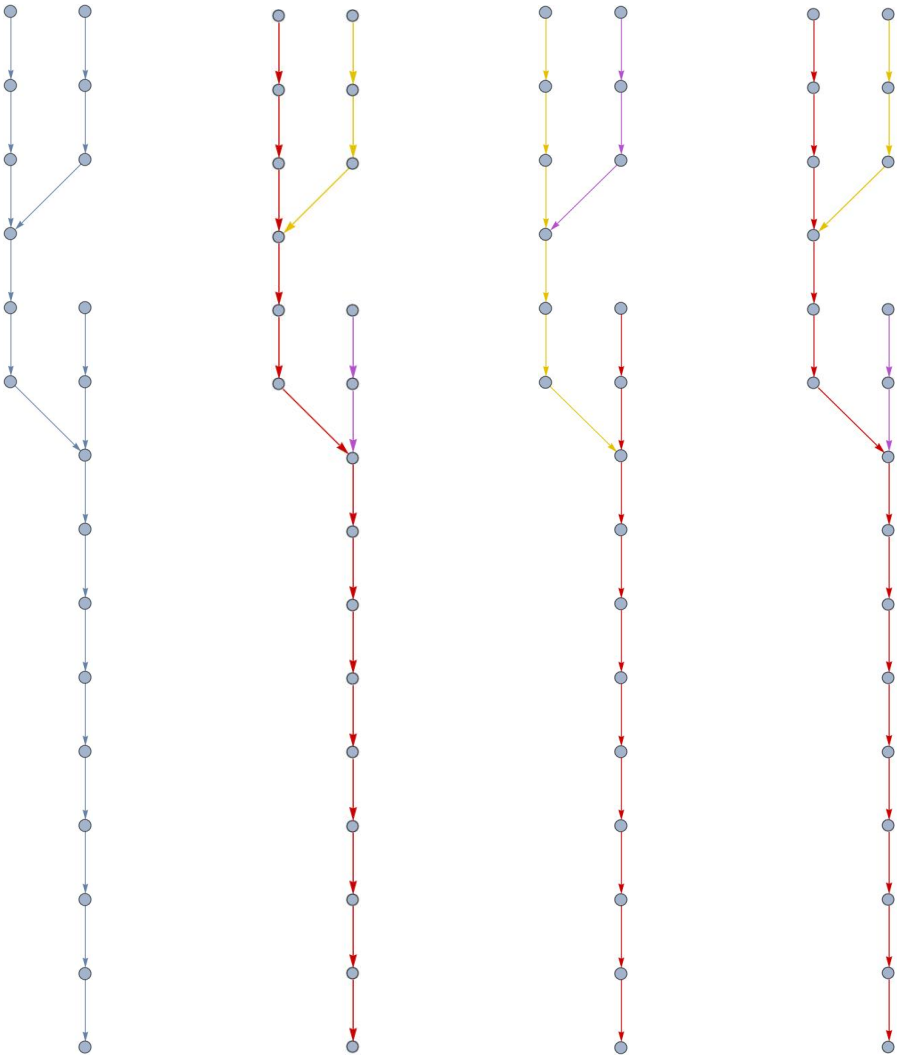


Figure 5.3: Solutions on the cat dataset, where each colour represents a different path. From left to right: the initial output graph, the long approximation solution, the close approximation solution and the flow approximation solution

5.8.2 *MNIST Dataset*

On the MNIST dataset for the IP, the long approximation yields an interestingness score of 360.132, the close approximation yields an interestingness score of 339.375 and the flow approximation also yields

an interestingness score of 339.014, where the optimal value is 360.132. The paths found are visually shown in Figure 5.4.

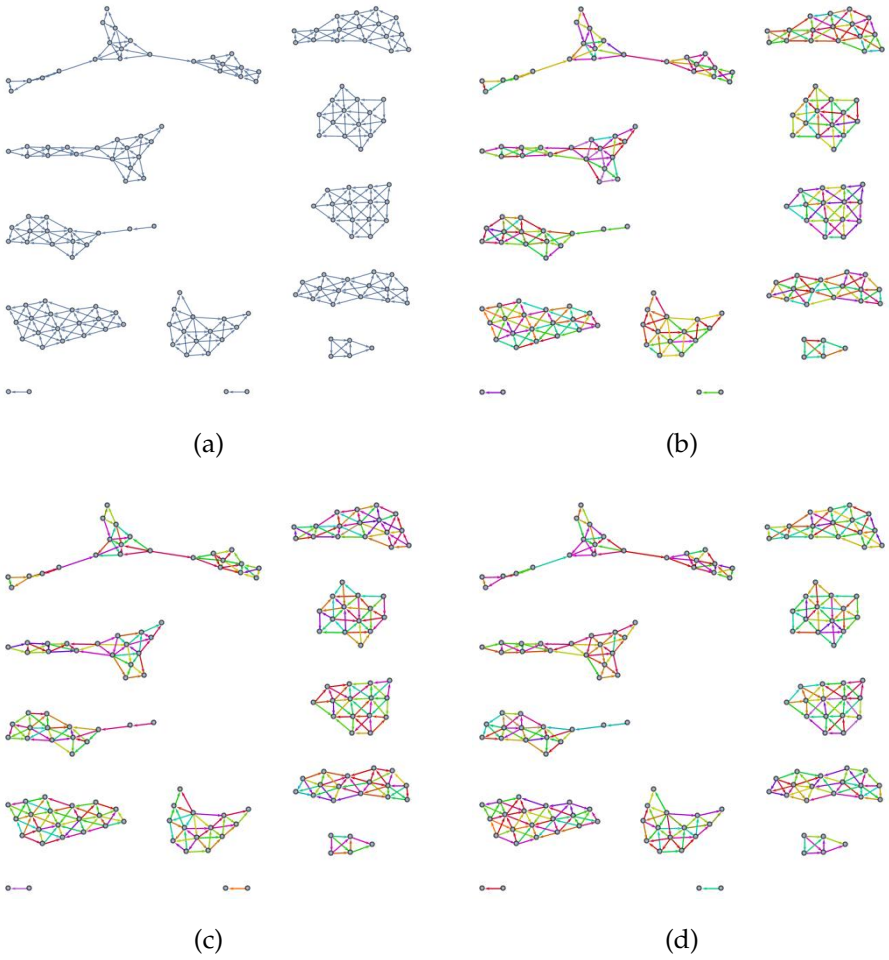
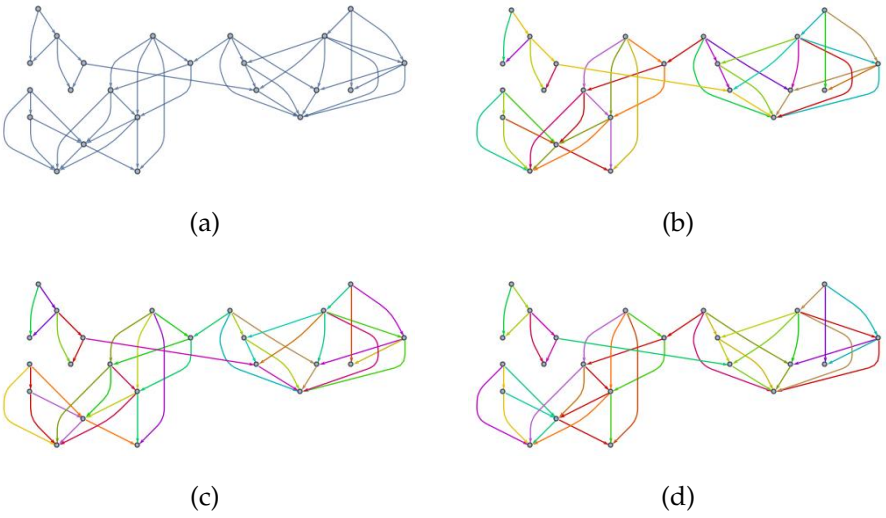


Figure 5.4: Solutions on the MNIST dataset: (a) original output graph, (b) long approximation, (c) close approximation, (d) flow approximation.

For easier visualization, we consider the largest weakly connected component of the output graph of the MNIST dataset. This component has 23 vertices and 45 edges, and is in the upper-left corner of any graph in Figure 4.18. On this partial MNIST dataset for the IP, the long approximation yields an interestingness score of 41.9309, the close approximation yields an interestingness score of 39.4708 and the flow approximation also yields an interestingness score of 40.6748, where the optimal value is 41.9309. The paths found are visually shown in Figure 5.5.

5.8.3 *Cancer Dataset*

On the cancer dataset for the IP, the long approximation yields an interestingness score of 407.933, the close approximation yields an

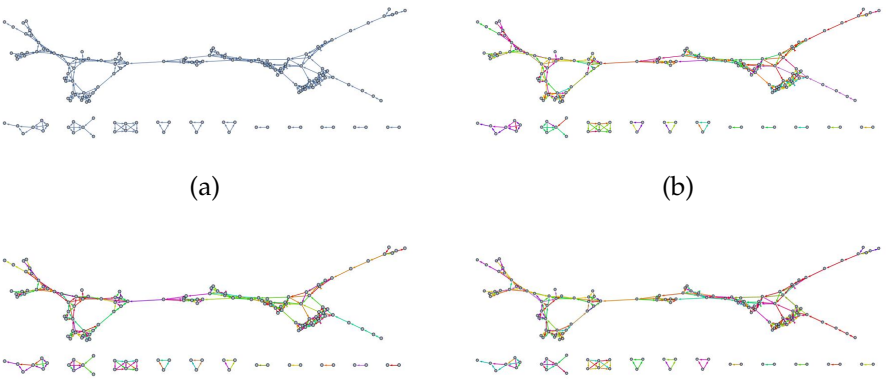


(a) (b)

(c) (d)

Figure 5.5: Solutions on the largest weakly connected component of the MNIST dataset: (a) original output graph, (b) long approximation, (c) close approximation, (d) flow approximation.

interestingness score of 367.701 and the flow approximation also yields an interestingness score of 364.729, where the optimal value is unknown. The paths found are visually shown in Figure 5.6.



(a) (b)

(c) (d)

Figure 5.6: Solutions on the cancer dataset: (a) original output graph, (b) long approximation, (c) close approximation, (d) flow approximation.

5.8.4 *Overview*

As for the dataset performance, we show in Table 5.1 the experimental interestingness score divided by the highest interestingness score found for the corresponding dataset.

Approximation Algorithm	Cat	MNIST	Cancer
Long	1	1	1
Close	0.889	0.942	0.901
Flow	1	0.941	0.894

Table 5.1: Performance of approximation algorithms for all datasets tested

5.9 DISCUSSION OF IP ALGORITHMS

Since the Mapper algorithm is intentionally an approximation of the dataset, it is coherent to propose approximation algorithms to its analysis.

Theoretically, the fastest algorithms are the long and close path approximations. These algorithms are heuristically motivated by opposite concerns. The longest path algorithm aims to greedily maximize the length of paths to maximize the interestingness score of each path. The close path algorithm aims to minimize the length of paths to avoid cutting potentially longer paths. While it is clearly either always choosing the longest path or choosing the closest path does not lead to the optimal solution, the longest path algorithm usually has the highest interestingness score.

The flow approximation algorithm was intended to bridge the gap between the longest path and close path approximation, but the results show this is the worst performing algorithm. This is a better performing algorithm than the close paths only in the case of sparse graphs with large paths. This shows further experimentation may be required to balance the performance of the longest path heuristic and close path heuristic.

A major difficulty in measuring the performance of approximation algorithms is finding the exact solution to IP of a given dataset, especially large or dense graph Mapper outputs. Dense graphs imply more possible paths, which makes any dense graph more difficult to solve than its sparse counterpart.

6 CONCLUSION

Our study of the complexity and design of multiple algorithms for interesting path problems allows their analysis for larger output graphs than ever before. Since the precision of the output graph depends on the size of the output graph, these algorithms enable practitioners to consider interesting subpopulations that may have otherwise been undetected by larger clusters.

While [41] mentions the IP problem as being the most general, the relaxation of the path length constraint simplifies the number of possible start and end vertices for any path, making in turn the IP problem less complex to approximate compared to the k-IP problem. Since the IP problem is not parametrized, this constitutes one step forward for the automatization of topological data mining.

6.1 NP-COMPLETENESS

6.1.1 *Max-IP Problem*

The Max-IP remains NP-complete in the general case, but we provide a way for practitioners to transform a general output graph of the Mapper algorithm into a directed acyclic graph using minor restrictions.

6.1.2 *k-IP Problem*

This work refutes the proof of NP-completeness found in [41] of the k-IP problem in the case of directed acyclic graphs for $k \geq 3$. It also proves the NP-completeness of the k-IP in the case of unweighted directed acyclic graphs for $k \geq 4$. Note that our NP-completeness proof is independent of the interestingness of the k-paths: we prove the more general NP-completeness of the disjoint k-paths maximization problem in unweighted directed acyclic graphs. We refer to this more general problem as the DAEDk problem.

6.1.3 *IP Problem*

The NP-completeness of the IP problem is proven in the case of weighted directed acyclic graphs, and a proposed construction may lead to the NP-completeness of the IP problem in the case of unweighted directed acyclic graphs, although this remains an open question. Note, even in the case of the IP problem, the weighting scheme

of interestingness used is not clearly required, as any tail-heavy path weighting function could possibly also lead to NP-completeness.

Overall, these proofs demonstrate the NP-completeness of the interesting path problems is highly dependent on the type of graph used.

6.1.4 *Unweighted IP is likely NP-complete*

While no proof is given of the NP-completeness of the unweighted IP on directed acyclic graphs, we conjecture the unweighted IP is NP-complete. A very similar proof to the weighted IP could be potentially used to prove the unweighted IP. Indeed, without loss of generality, it is possible to consider each weighted edge as a collection of edges with a total proportional to the weight attributed to the edge. This solution, however, likely requires many more edges per subgraph Γ of G, but still a constant number.

Here is an illustration of how we think we could handle the weights of the edges of each Γ_i. If for each weight w the equation $w = \log_2(l!)$ is solved, where l is the number of unweighted edges needed to represent w, one may prove the NP-completeness of the unweighted IP. For example, using $l_1 = 27308$ red edges, $l_2 = 76274$ yellow edges, $l_3 = 27425$ purple edges, $l_4 = 9021$ orange edges and $l_5 = 96789$ edges, we satisfy most of the constraints necessary for our proof of NP-completeness.

1. Red score: $\log_2(l_1!) \approx 363051$ instead of 363052
2. Yellow edge: $\log_2(l_2!) \approx 1127049$ instead of 1127044
3. Purple edge: $\log_2(l_3!) \approx 364775$ instead of 364776
4. Orange edge: $\log_2(l_4!) \approx 105521$ instead of 105519
5. Green edge: $\log_2(l_5!) \approx 1463445.78$ instead of 1463459

This is an avenue for future research.

6.2 APPLICATION

While the NP-completeness of these problems makes it impractical for practitioners to detect flares exactly on large output graphs, approximations allow for a significantly lower time complexity.

For studying or removing the central data path (Max-IP problem), practitioners may consider the Max-IP dynamic algorithm useful, especially for subsequent visual analysis of the output graph of the Mapper algorithm.

For detecting flares of a predetermined size (k-IP problem) with no concern for approximation bound, our experimental results show the

high performance of the linear programming approximation. If a good approximation bound is necessary, we recommend the use of tests using the colour-coding approximation with a compute time relative to the user. Alternatively, a random choice approach to choosing the highest value paths in the linear programming algorithm may also get better results.

For automatic flare detection (IP problem), the proof of approximation for minimum-path heuristics allows for creative approaches to flare detection. In our experiments, while the "long paths" heuristic had on average a better interestingness score than the other heuristics, no heuristic had a better performance than all the other heuristics in all cases.

Note, while these algorithms allow greater flare detection, solutions with better total interesting scores may not always indicate a better quality of flares. Considering this, in the case of the IP problem, practitioners should consider exploring domain-specific path searching heuristics that maintain the minimum-path property, such that the flares detected are more appropriate to the domain of the dataset.

6.3 FUTURE WORK

It remains an open question if the directed acyclic edge-disjoint k-path partition is NP-complete if the start and end vertices of each path are given.

The proofs and algorithms of this work rarely rely on the logarithmic nature of the interestingness scores for data mining. As such, alternative versions of interestingness may be considered using similar if not the same proofs and algorithms.

The linear programming approximation used to solve the k-IP has one of the best time complexity and the best approximation accuracy performance. However, whether the approximation bound is better than $1/k$ remains to be proven.

Overall, the k-IP problem and the IP problem follow the intention in the design of the Mapper algorithm, which is to analyze flares in complex datasets, but do so by analyzing the entire graph. It may be useful to consider designing other problems based on interestingness where the goal is to detect the main group of interesting flares without having to partition the entire graph.

Part I

APPENDIX

A

SOURCE

The source code of experiments, implementations, and figures is found at https://github.com/MarkCycVic/EfficientInterestingPaths. Note that one may use the figure generators to gain a more profound understanding of the structures created for NP-completeness. All results may be reproduced easily, except for the random graphs experiments. The latter required multiple machines and days to run, your results may vary.

BIBLIOGRAPHY

[1] Dustin L. Arendt, Matthew Broussard, Bala Krishnamoorthy, and Nathaniel Saul. *Cover Filtration and Stable Paths in the Mapper*. 2020. URL: https://openreview.net/forum?id=SJx0oAEYwH.

[2] Dustin L. Arendt, Matthew Broussard, Bala Krishnamoorthy, and Nathaniel Saul. *Steinhaus Filtration and Stable Paths in the Mapper*. 2020. arXiv: 1906.08256 [cs.LG].

[3] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer Publishing Company, Incorporated, 2013. ISBN: 3642635814.

[4] Ann S. Blevins, Dani S. Bassett, Ethan K. Scott, and Gilles C. Vanwalleghem. "From calcium imaging to graph topology." In: *Network Neuroscience* 6.4 (Oct. 2022), pp. 1125–1147. ISSN: 2472-1751. DOI: 10.1162/netn_a_00262. eprint: https://direct.mit.edu/netn/article-pdf/6/4/1125/2059765/netn_a_00262.pdf. URL: https://doi.org/10.1162/netn_a_00262.

[5] Cristian Bodnar, Cătălina Cangea, and Pietro Liò. *Deep Graph Mapper: Seeing Graphs through the Neural Lens*. 2020. arXiv: 2002.03864 [cs.LG].

[6] Peer-Timo Bremer, Ingrid Hotz, Valerio Pascucci, and Ronald Peikert. *Topological Methods in Data Analysis and Visualization III Theory, Algorithms, and Applications*. 1st ed. 2014. Mathematics and Visualization. Cham, Switzerland: Springer International Publishing, 2014. ISBN: 3-319-04099-5. DOI: 10.1007/978-3-319-04099-8. URL: https://doi.org/10.1007/978-3-319-04099-8.

[7] Giulia Campi, Giovanna Nicora, Giulia Fiorentino, Andrew Smith, Fulvio Magni, Silvia Garagna, Maurizio Zuccotti, and Riccardo Bellazzi. "A Topological Data Analysis Mapper of the Ovarian Folliculogenesis Based on MALDI Mass Spectrometry Imaging Proteomics." In: *Artificial Intelligence in Medicine*. Ed. by Allan Tucker, Pedro Henriques Abreu, Jaime Cardoso, Pedro Pereira Rodrigues, and David Riaño. Cham, Switzerland: Springer International Publishing, 2021, pp. 43–47. ISBN: 978-3-030-77211-6.

[8] Chandra Chekuri and Sanjeev Khanna. "Edge-Disjoint Paths Revisited." In: *ACM Trans. Algorithms* 3.4 (Nov. 2007), 46–es. ISSN: 1549-6325. DOI: 10.1145/1290672.1290683. URL: https://doi-org.proxy.library.carleton.ca/10.1145/1290672.1290683.

[9] Daniel H. Chitwood, Mitchell Eithun, Elizabeth Munch, and Tim Ophelders. "Topological Mapper for 3D Volumetric Images." In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by Bernhard Burgeth, Andreas Kleefeld, Benoît Naegel, Nicolas Passat, and Benjamin Perret. Cham, Switzerland: Springer International Publishing, 2019, pp. 84–95. ISBN: 978-3-030-20867-7.

[10] Michael B. Cohen, Yin Tat Lee, and Zhao Song. "Solving Linear Programs in the Current Matrix Multiplication Time." In: *CoRR* abs/1810.07896 (2018). arXiv: 1810.07896. URL: http://arxiv.org/abs/1810.07896.

[11] Claudio Contardo and Alain Hertz. "An exact algorithm for a class of geometric set-cover problems." In: *Discrete Applied Mathematics* 300 (2021), pp. 25–35. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2021.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0166218X21001864.

[12] Marek Cygan. "Improved approximation for 3-dimensional matching via bounded pathwidth local search." In: *CoRR* abs/1304.1424 (2013). arXiv: 1304.1424. URL: http://arxiv.org/abs/1304.1424.

[13] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research." In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[14] Tamal K. Dey and Yusu Wang. "Reeb Graphs: Approximation and Persistence." In: *Discrete & Computational Geometry* 49.1 (Jan. 2013), pp. 46–73. ISSN: 1432-0444. DOI: 10.1007/s00454-012-9463-z. URL: https://doi.org/10.1007/s00454-012-9463-z.

[15] Pawel Dlotko, Wanling Qiu, and Simon Rudkin. *Topological Data Analysis Ball Mapper for Finance*. 2022. arXiv: 2206.03622 [q-fin.GN].

[16] Pawel Dlotko and Simon Rudkin. *Visualising the Evolution of English Covid-19 Cases with Topological Data Analysis Ball Mapper*. 2020. arXiv: 2004.03282 [physics.soc-ph].

[17] Dorit Dor and Michael Tarsi. "Graph Decomposition is NP-Complete: A Complete Proof of Holyer's Conjecture." In: *SIAM Journal on Computing* 26.4 (1997), pp. 1166–1187. DOI: 10.1137/S0097539792229507. eprint: https://doi.org/10.1137/S0097539792229507. URL: https://doi.org/10.1137/S0097539792229507.

[18] Harish Doraiswamy and Vijay Natarajan. "Efficient algorithms for computing Reeb graphs." In: *Computational Geometry: Theory and Applications* 42.6 (2009), pp. 606–616. ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2008.12.003.

URL: https://www.sciencedirect.com/science/article/pii/S0925772108001193.

[19] Herbert Edelsbrunner and John Harer. *Computational topology : an introduction*. Providence, R.I: American Mathematical Society, 2010. ISBN: 9780821849255.

[20] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. "Topological persistence and simplification." In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. 2000, pp. 454–463. DOI: 10.1109/SFCS.2000.892133.

[21] Thomas Erlebach. *Approximation algorithms for edge-disjoint paths and unsplittable flow*. Berlin: Springer, 2006, pp. 97–134.

[22] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From Data Mining to Knowledge Discovery in Databases." In: *AI Magazine* 17.3 (1996), pp. 37–54. DOI: https://doi.org/10.1609/aimag.v17i3.1230. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1609/aimag.v17i3.1230. URL: https://onlinelibrary.wiley.com/doi/abs/10.1609/aimag.v17i3.1230.

[23] Martin Fürer and Huiwen Yu. "Approximating the k-Set Packing Problem by Local Improvements." In: *Combinatorial Optimization*. Ed. by Pierre Fouilhoux, Luis Eduardo Neves Gouveia, A. Ridha Mahjoub, and Vangelis T. Paschos. Cham, Switzerland: Springer International Publishing, 2014, pp. 408–420. ISBN: 978-3-319-09174-7.

[24] Harold N. Gabow. *A Data Structure for Nearest Common Ancestors with Linking*. 2016. arXiv: 1611.07055 [cs.DS].

[25] Cristian González García and Eva Álvarez Fernández. "What Is (Not) Big Data Based on Its 7Vs Challenges: A Survey." In: *Big Data and Cognitive Computing* 6.4 (2022). ISSN: 2504-2289. DOI: 10.3390/bdcc6040158. URL: https://www.mdpi.com/2504-2289/6/4/158.

[26] Tristan Gowdridge, Nikolaos Dervilis, and Keith Worden. *On topological data analysis for structural dynamics: an introduction to persistent homology*. 2022. arXiv: 2209.05134 [stat.ML].

[27] Lynn Greiner. "What is Data Analysis and Data Mining?" In: *Database Trends and Applications* (Jan. 7, 2011). URL: https://www.dbta.com/Editorial/Trends-and-Applications/What-is-Data-Analysis-and-Data-Mining-73503.aspx (visited on 10/11/2022).

[28] Jonathan L. Gross and Jay Yellen. *Graph theory and its applications*. 2nd ed. Discrete mathematics and its applications. Boca Raton: Chapman & Hall/CRC, 2006, p. 85. ISBN: 158488505X.

[29] Naoki Hamada and Kazuhisa Chiba. "Extraction and Visualization of Human Experts' Knowledge in Airplane Design via Mapper: Design Informatics Approach with Topological Data Analysis." In: *Transactions of Visualization Society of Japan* 41.162 (2021), pp. 29–30. ISSN: 0916-4731. DOI: 10.3154/jvs.41.162_29. URL: https://doi.org/10.3154/jvs.41.162_29.

[30] Frank Harary. "Sum graphs over all the integers." In: *Discrete Mathematics* 124.1-3 (1994), pp. 99–105. ISSN: 0012-365X. DOI: 10.1016/0012-365X(92)00054-U. URL: https://doi.org/10.1016/0012-365X(92)00054-U.

[31] Juris Hartmanis. "Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson)." In: *SIAM Review* 24.1 (1982), pp. 90–91. DOI: 10.1137/1024022. eprint: https://doi.org/10.1137/1024022. URL: https://doi.org/10.1137/1024022.

[32] Ian Holyer. "The NP-Completeness of Some Edge-Partition Problems." In: *SIAM Journal on Computing* 10.4 (Nov. 1981), pp. 713–5. URL: https://proxy.library.carleton.ca/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fscholarly-journals%2Fnp-completeness-some-edge-partition-problems%2Fdocview%2F920003959%2Fse-2%3Faccountid%3D9894.

[33] Jingliang Hu, Danfeng Hong, and Xiao Xiang Zhu. "MIMA: MAPPER-Induced Manifold Alignment for Semi-Supervised Fusion of Optical Image and Polarimetric SAR Data." In: *IEEE Transactions on Geoscience and Remote Sensing* 57.11 (2019), pp. 9025–9040. DOI: 10.1109/TGRS.2019.2924113.

[34] Wadah Ibrahim et al. "Visualization of exhaled breath metabolites reveals distinct diagnostic signatures for acute cardiorespiratory breathlessness." In: *Science Translational Medicine* 14.671 (2022), eabl5849. DOI: 10.1126/scitranslmed.abl5849. eprint: https://www.science.org/doi/pdf/10.1126/scitranslmed.abl5849. URL: https://www.science.org/doi/abs/10.1126/scitranslmed.abl5849.

[35] Wolfram Research, Inc. *Mathematica, Version 14.0*. Champaign, Illinois, 2024. URL: https://www.wolfram.com/mathematica.

[36] Shinsei Isojima, Kensuke Tanioka, Tomoyuki Hiroyasu, and Satoru Hiwa. "Preliminary Investigation of the Association Between Driving Pleasure and Brain Activity with Mapper-based Topological Data Analysis." In: *International Journal of Intelligent Transportation Systems Research* 21.3 (Dec. 2023), pp. 424–436. ISSN: 1868-8659. DOI: 10.1007/s13177-023-00371-3. URL: https://doi.org/10.1007/s13177-023-00371-3.

[37] Alon Itai, Yehoshua Perl, and Yossi Shiloach. "The complexity of finding maximum disjoint paths with length constraints." In: *Networks* 12.3 (1982), pp. 277–286. DOI: https://doi.org/10.1002/net.3230120306. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230120306. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230120306.

[38] Rachel Jeitziner, Mathieu Carrière, Jacques Rougemont, Steve Oudot, Kathryn Hess, and Cathrin Brisken. "Two-Tier Mapper, an unbiased topology-based clustering method for enhanced global gene expression analysis." In: *Bioinformatics* 35.18 (Feb. 2019), pp. 3339–3347. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz052. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/18/3339/48975370/bioinformatics_35_18_3339.pdf. URL: https://doi.org/10.1093/bioinformatics/btz052.

[39] Arthur B. Kahn. "Topological Sorting of Large Networks." In: *Commun. ACM* 5.11 (Nov. 1962), 558–562. ISSN: 0001-0782. DOI: 10.1145/368996.369025. URL: https://doi.org/10.1145/368996.369025.

[40] Ananth Kalyanaraman. "Scalable Topological Data Analysis for Life Science Applications." In: *Proceedings of the 18th ACM International Conference on Computing Frontiers*. CF '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 208. ISBN: 9781450384049. DOI: 10.1145/3457388.3459983. URL: https://doi.org/10.1145/3457388.3459983.

[41] Ananth Kalyanaraman, Methun Kamruzzaman, and Bala Krishnamoorthy. *Interesting Paths in the Mapper*. 2018. arXiv: 1712.10197 [cs.CG].

[42] Md Kamruzzaman. "Topological Data Analysis for Computational Phenomics: Algorithms and Applications." PhD thesis. 2020, p. 130. ISBN: 9798698542384. URL: https://proxy.library.carleton.ca/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fdissertations-theses%2Ftopological-data-analysis-computational-phenomics%2Fdocview%2F2467036564%2Fse-2%3Faccountid%3D9894.

[43] Methun Kamruzzaman, Ananth Kalyanaraman, Bala Krishnamoorthy, Stefan Hey, and Patrick S. Schnable. "Hyppo-X: A Scalable Exploratory Framework for Analyzing Complex Phenomics Data." In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 18.4 (2021), pp. 1535–1548. DOI: 10.1109/TCBB.2019.2947500.

[44] Viggo Kann. "Maximum bounded 3-dimensional matching is MAX SNP-complete." In: *Information Processing Letters* 37.1 (1991), pp. 27–35. ISSN: 0020-0190. DOI: https://doi.org/

10.1016/0020-0190(91)90246-E. URL: https://www.sciencedirect.com/science/article/pii/002001909190246E.

[45] Richard M. Karp. "Reducibility among Combinatorial Problems." In: *Complexity of Computer Computations*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9.

[46] Gideon Klaila, Vladimir Vutov, and Anastasios Stefanou. "Supervised topological data analysis for MALDI mass spectrometry imaging applications." In: *BMC Bioinformatics* 24 (July 2023), NA. ISSN: 14712105. URL: https://link-gale-com.proxy.library.carleton.ca/apps/doc/A756752137/AONE?u=ocul_carleton&sid=bookmark-AONE&xid=6c287a1e.

[47] Jon Michael Kleinberg and Michel X. Goemans. "Approximation Algorithms for Disjoint Paths Problems." AAI0597431. PhD thesis. USA, 1996.

[48] Joshua Levy, Christian Haudenschild, Clark Barwick, Brock Christensen, and Louis Vaickus. "Topological Feature Extraction and Visualization of Whole Slide Images using Graph Neural Networks." In: *Biocomputing 2021*, pp. 285–296. DOI: 10.1142/9789811232701_0027. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9789811232701_0027. URL: https://www.worldscientific.com/doi/abs/10.1142/9789811232701_0027.

[49] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. "Finding disjoint paths with different path-costs: Complexity and algorithms." In: *Networks* 22.7 (1992), pp. 653–667. DOI: https://doi.org/10.1002/net.3230220705. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230220705. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230220705.

[50] Ling-Yan Ma, Tao Feng, Chengzhang He, Mujing Li, Kang Ren, and Junwu Tu. "A progression analysis of motor features in Parkinson's disease based on the mapper algorithm." In: *Frontiers in Aging Neuroscience* 15 (2023). ISSN: 1663-4365. DOI: 10.3389/fnagi.2023.1047017. URL: https://www.frontiersin.org/articles/10.3389/fnagi.2023.1047017.

[51] "Najdorf Variation." In: *Modern Chess Openings*. 14th ed. Mckay Chess Library. David McKay Company, 1999, 244–267.

[52] Isaac Kofi Nti, Juanita Ahia Quarcoo, Justice Aning, and Godfred Kusi Fosu. "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects." In: *Big Data Mining and Analytics* 5.2 (2022), pp. 81–97. DOI: 10.26599/BDMA.2021.9020028.

[53] James B. Orlin. "Max Flows in O(Nm) Time, or Better." In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '13. New York, NY, USA: Association for Computing Machinery, 2013, 765–774. ISBN: 9781450320290. DOI: 10.1145/2488608.2488705. URL: https://doi.org/10.1145/2488608.2488705.

[54] Sriram V. Pemmaraju and Steven S. Skiena. *Computational discrete mathematics : combinatorics and graph theory with Mathematica*. Cambridge: Cambridge University Press, 2003, p. 231. ISBN: 0521806860.

[55] Georges Reeb. "Sur les points singuliers d'une forme de Pfaff completement integrable ou d'une fonction numerique [On the Singular Points of a Completely Integrable Pfaff Form or of a Numerical Function]." In: *Comptes Rendus Acad. Sciences Paris* 222 (1946), pp. 847–849. URL: https://cir.nii.ac.jp/crid/1571417125676878592.

[56] Samuel Rey, T. Mitchell Roddenberry, Santiago Segarra, and Antonio G. Marques. "Enhanced Graph-Learning Schemes Driven by Similar Distributions of Motifs." In: *IEEE Transactions on Signal Processing* 71 (2023), pp. 3014–3027. DOI: 10.1109/TSP.2023.3303639.

[57] Yuki Sato, Hiroki Kobayashi, Changyoung Yuhn, Atsushi Kawamoto, Tsuyoshi Nomura, and Noboru Kikuchi. "Topology optimization of locomoting soft bodies using material point method." In: *Structural and Multidisciplinary Optimization* 66.3 (Feb. 2023), p. 50. ISSN: 1615-1488. DOI: 10.1007/s00158-023-03502-2. URL: https://doi.org/10.1007/s00158-023-03502-2.

[58] Yossi Shiloach. *The two paths problem is polynomial*. Tech. rep. Stanford, CA, USA, 1978.

[59] Yossi Shiloach. "A Polynomial Solution to the Undirected Two Paths Problem." In: *J. ACM* 27.3 (July 1980), 445–456. ISSN: 0004-5411. DOI: 10.1145/322203.322207. URL: https://doi.org/10.1145/322203.322207.

[60] Yossi Shiloach and Yehoshua Perl. "Finding Two Disjoint Paths Between Two Pairs of Vertices in a Graph." In: *J. ACM* 25.1 (Jan. 1978), 1–9. ISSN: 0004-5411. DOI: 10.1145/322047.322048. URL: https://doi.org/10.1145/322047.322048.

[61] Yoshihisa Shinagawa, Tosiyasu L. Kunii, and Yannick L. Kergosien. "Surface coding based on Morse theory." In: *IEEE Computer Graphics and Applications* 11.5 (1991), pp. 66–78. DOI: 10.1109/38.90568.

[62] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. "Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition." In: *Eurographics Symposium on Point-Based Graphics*. Ed. by M. Botsch, R. Pajarola, B. Chen, and M. Zwicker. The Eurographics Association, 2007. ISBN: 978-3-905673-51-7. DOI: 10.2312/SPBG/SPBG07/091-100.

[63] Wai Shing Tang, Gabriel Monteiro da Silva, Henry Kirveslahti, Erin Skeens, Bibo Feng, Timothy Sudijono, Kevin K. Yang, Sayan Mukherjee, Brenda Rubenstein, and Lorin Crawford. "A topological data analytic approach for discovering biophysical signatures in protein dynamics." In: *PLoS Computational Biology* 18 (May 2022), e1010045. ISSN: 1553734X. URL: https://link-gale-com.proxy.library.carleton.ca/apps/doc/A705775544/AONE?u=ocul_carleton&sid=bookmark-AONE&xid=f8a12616.

[64] Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Anibal M. Medina-Mardones, Alberto Dassatti, and Kathryn Hess. "giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration." In: *J. Mach. Learn. Res.* 22.1 (Jan. 2021). ISSN: 1532-4435.

[65] Alpay Tekin, Ahmed Nebli, and Islem Rekik. *Recurrent Brain Graph Mapper for Predicting Time-Dependent Brain Graph Evaluation Trajectory*. 2021. arXiv: 2110.11237 [q-bio.NC].

[66] Jim Thatcher, David Retchless, Courtney Thatcher, and Kristine Jones. "Putting mapper on a map: cartographic visualizations of topological data analysis." In: *Abstracts of the ICA* 3 (Dec. 2021). ISSN: 25702106. URL: https://link-gale-com.proxy.library.carleton.ca/apps/doc/A686608787/AONE?u=ocul_carleton&sid=bookmark-AONE&xid=7de524c5.

[67] Theophile Thiery and Justin Ward. "An Improved Approximation for Maximum Weighted k-Set Packing." In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1138–1162. DOI: 10.1137/1.9781611977554.ch42. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611977554.ch42. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611977554.ch42.

[68] Pei Yee Tiew et al. "Sensitisation to recombinant Aspergillus fumigatus allergens and clinical outcomes in COPD." In: *European Respiratory Journal* 61.1 (2023). ISSN: 0903-1936. DOI: 10.1183/13993003.00507-2022. eprint: https://erj.ersjournals.com/content/61/1/2200507.full.pdf. URL: https://erj.ersjournals.com/content/61/1/2200507.

[69] Hendrik Jacob van Veen, Nathaniel Saul, David Eargle, and Sam W. Mangham. *Kepler Mapper: A flexible Python implementation of the Mapper algorithm*. Version 1.4.1. Oct. 2020. DOI: 10.5281/

zenodo.4077395. URL: https://doi.org/10.5281/zenodo.4077395.

[70] Laura J van 't Veer, Hongyue Dai, de V. Marc J van, Yudong D. He, and et al. "Gene expression profiling predicts clinical outcome of breast cancer." In: *Nature* 415.6871 (Jan. 2002), pp. 530–6. URL: https://proxy.library.carleton.ca/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fscholarly-journals%2Fgene-expression-profiling-predicts-clinical%2Fdocview%2F204517501%2Fse-2%3Faccountid%3D9894.

[71] William Wolberg, Olvi Mangasarian, Nick Street, and Street W. *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DW2B. 1995.

[72] Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. "On Variational Learning of Controllable Representations for Text without Supervision." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 10534–10543. URL: https://proceedings.mlr.press/v119/xu20a.html.

[73] Xiangyu Zhang, Kexin Zhang, and Yongjin Lee. "Machine Learning Enabled Tailor-Made Design of Application-Specific Metal–Organic Frameworks." In: *ACS Applied Materials & Interfaces* 12.1 (2020). PMID: 31820913, pp. 734–743. DOI: 10.1021/acsami.9b17867. eprint: https://doi.org/10.1021/acsami.9b17867. URL: https://doi.org/10.1021/acsami.9b17867.

[74] Yong Zhong, Liang Chen, Changlin Dan, and Amin Rezaeipanah. "A systematic survey of data mining and big data analysis in internet of things." In: *The Journal of Supercomputing* 78 (June 2022). DOI: 10.1007/s11227-022-04594-1.

[75] Youjia Zhou, Nithin Chalapathi, Archit Rathore, Yaodong Zhao, and Bei Wang. "Mapper Interactive: A Scalable, Extendable, and Interactive Toolbox for the Visual Exploration of High-Dimensional Data." In: *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*. 2021, pp. 101–110. DOI: 10.1109/PacificVis52677.2021.00021.

[76] Youjia Zhou, Nithin Chalapathi, Archit Rathore, Yaodong Zhao, and Bei Wang. *Mapper Interactive: A Scalable, Extendable, and Interactive Toolbox for the Visual Exploration of High-Dimensional Data*. 2021. arXiv: 2011.03209 [cs.CG].

[77] Youjia Zhou, Methun Kamruzzaman, Patrick Schnable, Bala Krishnamoorthy, Ananth Kalyanaraman, and Bei Wang. *PhenoMapper: An Interactive Toolbox for the Visual Exploration of Phenomics Data*. 2021. arXiv: 2106.13397 [cs.CG].

[78] Hein van der Holst and José Coelho de Pina. "Length-bounded disjoint paths in planar graphs." In: *Discrete Applied Mathematics* 120.1 (2002). Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization, pp. 251–261. ISSN: 0166-218X. DOI: https://doi.org/10.1016/S0166-218X(01)00294-3. URL: https://www.sciencedirect.com/science/article/pii/S0166218X01002943.