INTERESTING PATHS IN THE MAPPER COMPLEX

*Ananth Kalyanaraman** *Methun Kamruzzaman** *Bala Krishnamoorthy*†

Abstract.

Given a high dimensional point cloud of data with functions defined on the points, the mapper algorithm produces a compact summary in the form of a simplicial complex connecting the points. We study the problem of quantifying the interestingness of subpopulations in a given mapper complex.

First, we create a weighted directed graph $G = (V, E)$ using the 1-skeleton of the mapper complex. We use the average values at the vertices of a target function (dependent variable) to direct the edges from low to high values, and assign the difference (high−low) as the weight of the edge. Covariation of the remaining h functions (independent variables) is captured by a h-bit binary signature assigned to the edge. An *interesting path* in G is a directed path whose edges all have the same signature. The interestingness score of such a path as a sum of its edge weights multiplied by a nonlinear function of their corresponding ranks, i.e., the depths of the edges along the path. Such a nonlinear function could model application use-cases where the growth in the dependent variable values is expected to be concentrated in specific intervals of a path.

Second, we study three optimization problems on this graph G to quantify interesting subpopulations. In the problem Max-IP, the goal is to find the most interesting path in G, i.e., an interesting path with the maximum interestingness score. For the case where G is a directed acyclic graph (DAG), we show that Max-IP can be solved in polynomial time.

In the more general problem IP, the goal is to find a collection of interesting paths that are edge-disjoint, and the sum of interestingness scores of all paths is maximized. We also study a variant of IP termed k-IP, where the goal is to identify a collection of edge-disjoint interesting paths each with k edges, and the total interestingness score of all paths is maximized. While k-IP can be solved in polynomial time for $k \leq 2$, we show k-IP is NP-complete for $k \geq 3$ even when G is a DAG. We develop heuristics for IP and k-IP on DAGs, which use the algorithm for Max-IP on DAGs as a subroutine.

We have released open source implementations of our algorithms to find interesting paths. We also present a detailed experimental evaluation of this software framework on a real-world maize plant phenomics data set. We use interesting paths identified on several mapper graphs to explain how the genotype and environmental factors influence the growth rate, both in isolation as well as in combinations.

*School of Electrical Engineering and Computer Science, Washington State University, Pullman, USA
†Department of Mathematics and Statistics, Washington State University, Vancouver, USA
{ananth,md.kamruzzaman,kbala}@wsu.edu

1 Introduction

Data sets from many applications come in the form of point clouds often in high dimensions along with multiple functions defined on these points. Topological data analysis (TDA) has emerged in the past two decades as a new field whose goal is to summarize such complex data sets, and facilitate the understanding of their underlying topological and geometric structure. In this paper, we focus on the *mapper* algorithm [26], a TDA method that has gained significant traction across various application domains [1, 9, 12, 14, 15, 22, 23, 24, 25, 28].

Starting from a point cloud X (typically sampled from a metric space), the mapper algorithm studies the topology of the sublevel sets of a *filter function* $f : X \to Z \subset \mathbb{R}$. Starting with a cover $\mathcal{Z}$ of Z, the mapper algorithm obtains a cover of the domain X by pulling back $\mathcal{Z}$ through f. This pullback cover is then refined into a connected cover by splitting each of its elements into various clusters using a clustering algorithm. A compact representation of the data set, termed the mapper complex, is obtained by taking the nerve of this connected cover—this is a simplicial complex with one vertex per cluster, one edge per pair of intersecting clusters, and one p-simplex per non-empty $(p+1)$-fold intersection in general. The method can naturally consider multiple filter functions $f_i : X \to Z_i$, with covers $\mathcal{Z}_i$ jointly pulled back to obtain the cover of X. For instance, with two filter functions f_1 and f_2, elements of such a joint cover of X are defined by pairwise intersections of the elements of the pullback covers of $\mathcal{Z}_1$ and $\mathcal{Z}_2$. Equivalently, one could consider them together as a *single* vector-valued filter function $\mathbf{f} : X \to Z \subset \mathbb{R}^h$ for $h \geq 2$.

The mapper algorithm has been used in a growing number of applications from diverse domains recently, ranging from medicine [9, 14, 22, 23, 24, 25, 28] to agricultural biotechnology [12] to basketball player profiles [1] to voting patterns [15]. It is also the main engine in the data analytics software platform of the firm Ayasdi. The key to all these success stories is the ability of the mapper algorithm to identify subsets of X, i.e., *subpopulations*, that behave distinctly from the rest of the points. In fact, this feature of the mapper algorithm distinguishes it from many traditional data analysis techniques based on, e.g., machine learning, where the goal is usually to identify patterns valid for the entire data set.

Several researchers have recently studied mathematical and foundational aspects of the mapper algorithm (see Section 1.3). These results have enabled robust ways to build *one* representative mapper complex for any given data set. One could follow the work of Carrière et al. [3], or identify parameters corresponding to a stable range in the persistence diagram of the multiscale mapper complex [5]. While a collection of mapper complexes built at multiple scales, e.g., the multiscale mapper complex [5], could provide a more detailed representation of the data, efficient summarization of the entire representation as well as the extraction of insights relevant to the application still remain challenging. Hence working with a single mapper complex could be considered the desirable setting from the point of view of most applications.

1.1 Interesting Paths in the Mapper Complex: Motivation

While the mapper complex is high dimensional in general, applications have typically used its 1-skeleton (referred to as the mapper graph) for interpretations. Features such as paths, flares, and loops, in the selected mapper graph often convey meaningful insights on the data set. But many applications demand more precise quantification of the features, as well as to track the corresponding subpopulations as they evolve along such features. For instance, in plant phenomics [12], we are interested in identifying specific varieties (i.e., genotypes) of a crop that show resilient growth rates as one or more environmental factors vary during the growing season. Each such "subpopulation" (e.g., a subset of varieties grown in a certain location or environment) with the associated factors would suggest a testable hypothesis for the practitioner.

Many applications also have characteristics where a performance variable (e.g., a behavioral trait) is measured/observed over a period of time or over a wide range of an environmental variable (or variables). For instance, in the plant scenario described above, the growth rate of a plant is expected to vary with time (since the day of planting). As another example, consider a hospital dataset where a key health indicator variable for patients, such as blood sugar, is expected to evolve with the type of treatment they receive in the hospital over a period of time. Such health indicators and their response patterns are also expected to vary from patient (groups) to patient (groups). Under these settings, how does one effectively track the evolution of behavior exhibited in different parts of the population as a function of potential environmental variables? The mapper algorithm provides a framework within which one could build an effective approach. More specifically, we could use time or an environmental variable as a filter function, and the behavioral trait of interest as the distance function (in our implementation of the mapper algorithm—see the start of Section 2). In the resulting mapper graph, a cluster would represent a subset of individuals who share similar behavioral traits under similar environments (or time periods). Further, a path would represent a trail of clusters that persist over an interval of filter function values. In addition, if one were to orient the edges along the path in the direction of increasing cluster means of the performance values, then a directed path would represent the trajectory of a subpopulation from a low end to a high end of the filter variable spectrum. Consequently, identifying such paths in the mapper complex can be insightful.

To further quantify such a path in the mapper complex, it is possible to define its "interestingness score". Such a score can be used to rank order the paths so that the user can examine them in the decreasing order of their interestingness. We first associate every edge with a weight that is the absolute difference in the cluster means of the performance values of the two clusters it connects. In Section 2.1, we provide a definition of the interestingness score of a path in the mapper graph as a sum of its edge weights multiplied by a nonlinear function of their corresponding ranks, i.e., the depths of the edges along the path. Such a nonlinear function could help model application use-cases where increases in the performance variable values are expected to be concentrated in specific intervals of a path. For example, consider plant growth accelerating later in the growth season or peaking as a cumulative temperature index such as the Growing Degree Days (GDD) reaches a certain value [18]. Similarly, consider a terminally ill patient's health condition deteriorating rapidly as mortality nears or after certain procedural complication arises. We further elaborate on our interestingness score in Section 2.1.

1.2 Our Contributions

We propose a framework for quantifying the interestingness of subpopulations in a given mapper complex. For the input point cloud X, we assume the mapper complex is constructed with h filter functions $f_i : X \to Z_i$ that represent independent variables, and a target function (i.e., a variable of interest or a dependent variable) $g : X \to Z$ that is distinct from the f_i's. We describe our implementation of the mapper algorithm in Section 2. The mapper complex could be a high-dimensional simplicial complex depending on the choice of covers $\mathcal{Z}_i$ for Z_i. But we concentrate on its 1-skeleton, as described below.

Formulation: We create a weighted directed graph $G = (V, E)$ using the 1-skeleton of the mapper complex, which we call the mapper graph. We use the average values of g at the vertices (i.e., clusters) to direct the edges from low to high values. We set the difference between the average values at the vertices (high$-$low) as the weight of the edge. Covariation of the h functions f_i is captured by a h-bit binary signature assigned to the edge. We define an *interesting path* in G as a directed path whose edges all have the same signature (all references to a "path" in this paper imply a simple path, i.e., no vertices are repeated). Further, we define the interestingness score of such a path as a sum of its edge weights multiplied by a nonlinear function of their corresponding ranks, i.e., the depths of the edges along the path. The goal is to value more the contribution from an edge deep in the path than that from a similar edge which appears at the start.

Theoretical Results: We study three optimization problems on this graph G to quantify interesting subpopulations. In the problem Max-IP, the goal is to find the most interesting path in G, i.e., an interesting path with the maximum interestingness score. We concentrate on the case where G is a directed acyclic graph (DAG), which is a typical setting in many applications. We show that Max-IP can be solved in polynomial time—in $O(m\delta d_{\text{in}})$ time and $O(mn)$ space where m and n are the number of edges and vertices respectively, and δ and d_{in} are the diameter of G and the maximum indegree of any vertex in G respectively. Note that $\delta < n$ and $d_{\text{in}} < n$ (as G is a DAG).

In the more general problem IP, the goal is to find a collection of interesting paths such that these paths form an exact cover of E and the overall sum of interestingness scores of all paths is maximum. The collection of paths identified by IP could include some short ones in terms of number of edges. Hence we study also a variant of IP termed k-IP, where the goal is to identify a collection of interesting paths each with k edges for a given number k, an edge in E is part of at most one such path, and the total interestingness score of all paths is maximum. While k-IP can be solved in polynomial time for $k \leq 2$, we show k-IP is NP-complete for $k \geq 3$. Finally, we develop heuristics for IP and k-IP on DAGs, which use the algorithm for Max-IP on DAGs as a subroutine, and run in $O(mnd_{\text{in}})$ and $O(mkd_{\text{in}})$ time for IP and k-IP, respectively.

In addition, we consider the more general setting where G is a directed graph. This setting could arise in practice when the average values of the target function g are too close for pairs of nodes in the mapper construction. Hence edges going both ways between them are included in G, making G no longer acyclic. We show that Max-IP is NP-complete in this general setting.

Software and Experimental Evaluation: In this paper, we focus more on the theoretical aspects of interesting paths. However, we are also actively developing and maintaining an open source software repository that integrates all our ongoing implementations, and their experimental evaluations and applications. Section 7 reports on this ongoing effort. In Section 8, we present details of our use of plant phenomics [10] as a novel application domain for test, validation, and discovery using interesting paths identified by our software. Phenomics is a nascent branch of modern biology whose core goals includes the understanding of how crop genotypes (G) interact with environments (E) to produce varying performance traits (phenotypes (P)) (denoted $G \times E \to P$) [16]). We use interesting paths identified on several mapper graphs built on a maize phenomics data set to explain how the genotype as well as environmental factors influence the growth rate, both in isolation as well as in combination. In particular, we study the effects of several filter functions representing environment (E) including time after planting, temperature, humidity, and solar radiation, as well as crop genotypes (G) on growth rate, which is the phenotype (P) of interest.

1.3 Related work

In most previous applications of the mapper algorithm [1, 9, 14, 15, 22, 23, 24, 25, 28], interesting subpopulations are characterized by features (paths, flares, loops) identified in a visual manner. As far as we are aware, our work proposes the first approach to rigorously quantify the interesting features, and to rank them in terms of their interestingness. Munch and Wang [20] showed convergence of the mapper complex to a categorified Reeb space associated with the data. Dey et al. [5] considered a *multiscale* mapper framework, and proved stability results for features within this framework using techniques from persistence. The works of Carrière et al. [3, 4] present a rigorous theoretical framework for the 1-dimensional mapper algorithm, where the features are identified as points in an extended persistence diagram. While addressing stability of features in the mapper complex under various rigorous settings, these lines of work do not address the relative importance of the features in the context of the application generating the data. Our work can be considered as a post processing of the mapper complex identified by the methods of Carrière et al. While our framework can naturally consider multiple filter functions for a given mapper complex, we do not address the stability of the interesting paths identified.

The interesting paths problems we study are related to the class of nonlinear shortest path and minimum cost flow problems previously investigated. Non-additive shortest paths have been studied [29], and the more general minimum concave cost network flow problem has been shown to be NP-complete [8, 31]. At the same time, versions of shortest path or minimum cost flow problems where the contribution of an edge depends nonlinearly on its position or depth in the path appear to have not received much attention. Hence the specific problems we study should be of independent interest as a new class of nonlinear longest path (equivalently, shortest path) problems.

2 Methods

We refer the reader to the original paper by Singh et al. [26] for background on the mapper algorithm, and other recent work [3, 4, 5] for related constructions. We describe the details of our implementation of the mapper algorithm, highlighting certain key differences from the default option.

Our Implementation of the mapper algorithm: We start with an input point cloud X in high-dimensional space. In the default setting, each dimension represents a filter function, or equivalently, all of them could be considered together as a high-dimensional filter function. In our implementation, we consider individual filter functions $f_i : X \to Z_i \subset \mathbb{R}$ for $i = 1, \ldots, h$. We build covers $\mathcal{Z}_i$ of Z_i by choosing the lengths and overlap of the intervals (resolution and gain). We then pullback the joint cover of all $\mathcal{Z}_i$ through f_i to obtain a cover of X. The default implementation considers a cover of product space $Z_i \times Z_2 \times \cdots \times Z_h$, and hence is somewhat more general than our implementation. For instance, it may not be obvious how to decompose a cover of the product into the product of covers of the individual Z_is. We distinguish another function $g : X \to \mathbb{R}$ from the filter functions. In the setting of a typical application, g could represent a target variable of interest to the application expert. Typically, this variable would represent a performance variable whose relationship to the independent variables (identified by f_i) is of interest. Given such a variable of choice for g, we use g to *refine* the pullback cover of X, i.e., to divide each cover element into connected components, which constitutes the clustering step within the mapper framework. This step is achieved by computing a clustering based on g alone. Note that in the default implementation of mapper, this clustering step is performed in all dimensions of X. Our use of g for clustering and a set of independent variables (f_i) for filtering, is motivated by applications where such a demarcation could help facilitate easier interpretability of the mapper results. This is demonstrated in detail via a concrete application case-study in Section 8.

In what follows, we denote the resulting mapper complex as M, and the directed graph constructed from its 1-skeleton as G (see Section 2.1.1).

2.1 Interesting Paths and Interestingness Scores

Each vertex in M represents a cluster of points from X that have similar values of function g, the dependent variable. An edge in M connects two such clusters containing a non-empty intersection of points. By definition, each edge in M connects clusters belonging to distinct elements of the pullback cover of X, and hence the corresponding values of the filter functions f_i also change when moving along the edge. Therefore, by following a trail of vertices (i.e., clusters) whose average g values are monotonically varying, we can capture subpopulations that gradually or abruptly alter their behavior as measured by g under continuously changing filter intervals. For instance, a plant scientist interested in crop resilience may seek to identify a subset of crop individuals/varieties that exhibit sustained or accelerated growth rates (g) despite potentially adverse fluctuations in temperature (f_1) and humidity (f_2). We formulate the problem of identifying such subpopulations as that of finding interesting edge-disjoint paths in a directed graph.

2.1.1 Graph Formulation

We construct a weighted directed graph $G = (V, E)$ representation of the 1-skeleton of M along with some additional information. We refer to this graph as the *mapper graph*. Let $n = |V|$ and $m = |E|$ denote the numbers of vertices and edges in G, respectively. We set V as the set of vertices (0-simplices) of M, and E as the set of edges (1-simplices) of M. We assign directions and weights to the edges as follows. Each vertex $u \in V$ denotes a subset of points from X that constitute a partial cluster. We denote this subset as $X(u)$. We let $g(u)$ and $f_i(u)$ denote the average values of the dependent variable g (used for clustering) and the filter function f_i, respectively, for all points in u:

$$g(u) = \frac{\Sigma_{x \in X(u)}\, g(x)}{|X(u)|} \quad \text{and} \quad f_i(u) = \frac{\Sigma_{x \in X(u)}\, f_i(x)}{|X(u)|}, \; i = 1, \ldots, h.$$

We let $\omega(u)$ represent the weight of a vertex u. In this paper, we set $\omega(u)$ to be equal to $g(u)$. For an edge $e = (u, v)$ in E, we assign as its weight as $\omega(e) = |\omega(u) - \omega(v)| = |g(u) - g(v)|$. Notice $\omega(e) \geq 0$ for all edges e in G.

Each edge is directed from the vertex with lower weight to the vertex with higher weight, as illustrated in Figure 1. If the vertex weights are equal, one of the two directions is chosen arbitrarily. With the edges directed in this fashion, the resulting graph is guaranteed to be a directed acyclic graph (DAG). Consequently, the optimization problems we address in this paper (see Section 2.1.2) will be presented for DAG inputs.

A more general approach would involve including both forward and backward edges in G between vertices whose weights are equal, or close to equal. The graph G need not be acyclic in this setting, and we consider certain implications of the optimization problems on directed graphs in Section 6.

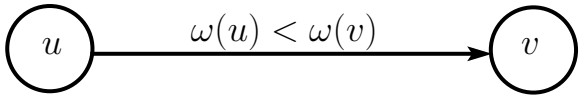


Figure 1: Direct edge from u to v if $\omega(v) > \omega(u)$, and from v to u if $\omega(v) < \omega(u)$. If the weights are equal, then one of the two directions is chosen arbitrarily.

We assign a h-bit binary signature $\mathrm{Sig}(e) = b_1 b_2 \ldots b_h$ to the oriented edge $e = (u, v)$ (i.e., $e : u \to v$) to capture the covariation of g and the filter functions f_i. We set $b_i = 1$ if $f_i(u) \leq f_i(v)$, and $b_i = 0$ otherwise.

Definition 2.1. An *interesting k-path* for a given k with $1 \leq k \leq n-1$ is a directed path $P = [e_{i_1}, \ldots, e_{i_k}]$ of k edges in G, such that $\mathrm{Sig}(e_r)$ is identical for all $r = i_1, \ldots, i_k$. An *interesting path* is a path of arbitrary length in the interval $[1, n-1]$.

Definition 2.2. Given an interesting k-path $P = [e_{i_1}, \ldots, e_{i_k}]$ in G as specified in Definition 2.1, we define its *interestingness score* as follows.

$$\mathcal{I}(P) = \sum_{r=1}^{k} \omega(e_{i_r}) \times \log(1 + r). \tag{1}$$

In particular, the contribution of an edge $e \in P$ to $\mathcal{I}(P)$ is set to $\omega(e) \times \log(1 + \text{rank}(e, P))$, where $\text{rank}(e, P)$ is the rank or order of edge e as it appears in P.

Intuitively, we use the rank of an edge as an inflation factor for its weight—the later an edge appears in the path, the more its weight will count toward the interestingness of the path. This logic incentivizes the growth of long paths when we try to maximize the interestingness score. The log function, on the other hand, helps temper this growth in terms of number of edges. Inflation of weights for edges that appear later in the path is motivated by the potential interpretation of interesting paths in the context of real world applications. For instance, while analyzing plant phenomics data sets [12, 16], we expect plants to show accelerated growth spurts later in the growth season. Plants showing such spurts later in the season are potentially more interesting to the practitioner than ones showing a steady growth rate throughout the season. Also, note that *every* path has an interestingness score, but our goal is to find path(s) with high (est) interestingness score(s).

Remark 2.3. The above framework can be modified easily to characterize robust interesting paths, where the signature matching condition is relaxed—for instance, $b_i = 1$ if $0.9 f_i(v) \leq f_i(u) \leq f_i(v)$.

Remark 2.4. While we assume g and f_i are functions from X to $\mathbb{R}$, our framework could handle more general functions as well. If some f_i is a vector-valued function, for instance, we could first compute pairwise distances of the points in X using f_i, and then assign to each point in X its average distance to all other points as a "surrogate" function.

Remark 2.5. The framework applies without change to cases where g is used along with other functions to cluster. In fact, g could be used also as a filter function as long as it is used for clustering.

2.1.2 Optimization Problems

We now present multiple optimization problems with the broader goal of identifying interesting path(s) that maximize interestingness score(s).

Max-IP: Find an interesting path P in G such that $\mathcal{I}(P)$ is maximized.

k-IP: For a given k between 1 and $n - 1$, find a collection $\mathcal{P}$ of interesting k-paths such that each $e \in E$ is part of at most one $P \in \mathcal{P}$, and the total interestingness score $\mathcal{I}(\mathcal{P}) = \sum_{P \in \mathcal{P}} \mathcal{I}(P)$ is maximized.

IP: Find a collection $\mathcal{P}$ of interesting paths in G such that the total interestingness score $\mathcal{I}(\mathcal{P}) = \sum_{P \in \mathcal{P}} \mathcal{I}(P)$ is maximized ($\mathcal{P}$ will exactly cover E, i.e., each $e \in E$ is part of exactly one $P \in \mathcal{P}$).

Both IP and k-IP produce edge-disjoint collections of interesting paths. In IP, every edge in G is part of an interesting path in $\mathcal{P}$. But this setting might include several short (in number of edges) interesting paths. In k-IP, each interesting path found has exactly k edges, and some edges in E might not be part of any interesting k-path in $\mathcal{P}$. Hence the paths identified by k-IP are likely to be more meaningful in practice.

Remark 2.6. The log(1+rank) factor in the interestingness score in Equation (1) makes each of the above optimization problems nonlinear. At the same time, the type of nonlinearity introduced here is distinct from the ones studied in the literature, e.g., in non-additive shortest paths [29], or in minimum concave cost flow [8, 31]. Hence these problems form a new class of nonlinear longest (equivalently, shortest) path problems, which would be of interest independent of their application in the context of the mapper algorithm and TDA.

Remark 2.7. Our directed graph formulation (in Section 2.1.1) with signatures determined by h filter functions could also be considered equivalently as the computation of the coboundary of the filter functions seen as a 0-cochain with coefficients in $\mathbb{R}^h$ [21]. Further, under appropriate assumptions on the filter functions being smooth, candidates for interesting paths could be seen as flows in a gradient field [30, §14]. Under these assumptions, one could argue that the graph constructed will necessarily be a DAG, and results from Morse theory [17, 19] would also apply. But we are not assuming the functions involved are necessarily smooth. Further, the $\log(1 + \text{rank})$ factor used in defining our interestingness score (in Equation 1) is not captured by default approaches for maximal flows in gradient fields or by Morse theory.

3 The Max-IP Problem

The goal of Max-IP is to identify an interesting path with the maximum interestingness score. We show Max-IP on a directed acyclic graph (DAG) can be solved in polynomial time.

3.1 Max-IP on Directed Acyclic Graphs

Lemma 3.1. Max-IP *on a directed acyclic graph* $G = (V, E)$ *is in* P.

Proof. We present a polynomial time algorithm for Max-IP on a DAG (as proof of Lemma 3.1). The input is a DAG $G = (V, E)$ with n vertices and m edges, with edge weights $\omega(e) \geq 0$ and signatures $\text{Sig}(e)$ for all $e \in E$. The output is an interesting path P^* which has the maximum interestingness score in G. We use dynamic programming, with the forward phase computing $\mathcal{I}(P^*)$ and the backtracking procedure reconstructing a corresponding P^*.

Let $T(i, j)$ denote the score of a maximum interesting path of length j edges ending at edge e_i for $i \in [1, m]$. Since an interesting path could be of length at most $(n-1)$, we have $j \in [1, n-1]$. Therefore the values in the recurrence can be maintained in a 2-dimensional table of size $m \times (n-1)$, as illustrated in Figure 2. The algorithm has three steps:

- **Initialization:** $T(i, 1) = \omega(e_i) \times \log(2)$, where $1 \leq i \leq m$.
- **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 $e' = (w, u)$ and $\text{Sig}(e') = \text{Sig}(e)$. Let $\text{Pred}(e)$ denote the set of all predecessor edges of e. Note that $\text{Pred}(e)$ can be possibly empty. We define the recurrence for $T(i, j)$ as follows.

$$T(i, j) = \max_{e_{i'} \in \text{Pred}(e_i)} \big\{ T(i', j-1) + \omega(e_i) \times \log(1 + j) \big\}. \tag{2}$$

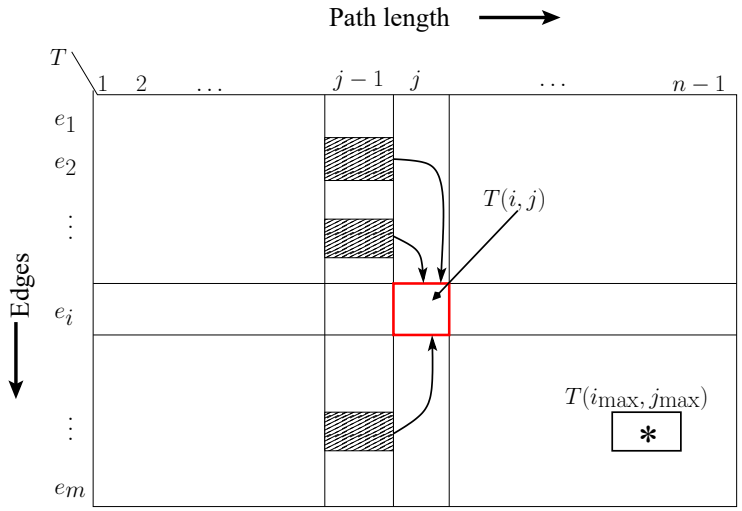


Figure 2: Table $T(i,j)$ for the MAX-IP algorithm.

Let e_i^* give the maximum in the above recurrence. We record this edge as the predecessor of e_i in the most interesting path, i.e., $\text{pred}(e_i) = e_i^*$.

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

Proof of Correctness: Any interesting path in G can be at most $n-1$ edges long. As a particular edge could appear anywhere along such a path, its rank can range between 1 and $n-1$. Hence the $m \times (n-1)$ recurrence table T sufficiently captures all possibilities for each edge in E. The following key observation completes the proof. Let $P^*(i,j)$ denote an optimal scoring path, if one exists, of length $j \in [1, n-1]$ ending at edge $e_i \in E$. If $P^*(i,j)$ exists and if $j > 1$, then there should also exist $P^*(i', j-1)$ where $i' \in \text{Pred}(e_i)$. Furthermore, the edge e_i *could not* have appeared in $P^*(i', j-1)$ because G is acyclic. Therefore, due to the edge-disjoint nature of $P^*(i', j-1)$ and the remainder of $P^*(i,j)$ (which is e_i), the principle of optimality is preserved—i.e., the maximum operator in Equation (2) is guaranteed to ensure optimality of $T(i,j)$.

Complexity Analysis: The above dynamic programming algorithm can be implemented to run in $O(mn)$ space and a worst-case time complexity of $O(mnd_{\text{in}})$, where d_{in} denotes the maximum indegree of any vertex in V. □

3.1.1 Algorithmic Improvements

The above dynamic programming algorithm for MAX-IP for DAGs can be implemented to run in space and time smaller in practice than the worst case limits. First, we note that computing the full table T is likely to be wasteful, as it is likely to be sparse in practice. The sparsity of T follows from the observation that an interesting path of length j ending at

edge e_i can exist only if there exists at least one other interesting path of length $j-1$ ending at one of e_i's predecessor edges. We can exploit this property by designing an iterative implementation as follows.

Instead of storing the entire table T, we store only the rows (edges), and introduce columns on a "need basis" by maintaining a dynamic list $L(e_i)$ of column indices for each edge e_i.

S1) Initially, we assign $L(e_i) = \{1\}$, as each edge is guaranteed to be in an interesting path of length at least 1 (the path consisting of the edge by itself).

S2) In general, the algorithm performs multiple iterations; within each iteration, we visit and update the dynamic lists for all edges in E as follows. For every edge $e_{i'} \in \text{Pred}(e_i)$, $L(e_i) = L(e_i) \cup \{\ell+1 \mid \ell \in L(e_{i'})\}$. The algorithm iterates until there is no further change in the lists for any of the edges.

The number of iterations in the above implementation can be bounded by the length of the longest path in the DAG (i.e., the diameter δ), which is less than n. Also, we implement the list update from predecessors to successors such that each edge is visited only a constant number of times (despite the varying products of in- and out-degrees at different vertices). To this end, we implement the update in S2 as a two-step process: first, performing a union of all lists from the predecessor edges of the form $(*, v)$ so that the merged lists can be used to update the lists of all the successor edges of the form $(v, *)$. Thus the work in each iteration is bounded by $O(m)$.

Taken together, even in the worst-case scenario of $(\delta + 1)$ iterations, the overall time to construct these dynamic lists is $O(m\delta)$. Furthermore, during the list construction process, if one were to carefully store the predecessor locations using pointers, then the computation of the $T(i, j)$ recurrence in each cell can be executed in time proportional to the number of *non-empty* predecessor values in the table. Overall, this revised algorithm can be implemented to run in time $O(m\delta d_{\text{in}})$, and in space proportional to the number of non-zero values in the matrix.

Further, the above implementation is also inherently parallel since the list value at an edge in the current iteration depends only on the list values of its predecessors from the previous iteration.

4 The k-IP Problem

The goal of k-IP for $k \leq n-1$ is to find a set $\mathcal{P}$ of edge-disjoint interesting k-paths such that the sum of their interestingness scores is maximized. We show that k-IP on directed acyclic graphs can be solved in polynomial time for $k \leq 2$ (see Lemma 4.1). On the other hand, we show that k-IP is NP-Complete for $k \geq 3$ (see Theorem 4.2).

The smallest value of k for which k-IP is nontrivial is 2. We can solve 2-IP as a weighted matching problem.

Lemma 4.1. k-IP *on directed acyclic graph* $G = (V, E)$ *is in* P *for* $k \leq 2$.

Proof. The case of $k = 1$ turns out to be trivial. An optimal solution for 1-IP is obtained by taking $\mathcal{P}$ as a collection of m interesting 1-paths each comprised of a single edge. These 1-paths are edge-disjoint by definition, and the need to compare signatures within a path does not arise. Since all edge weights $\omega \geq 0$, the total interestingness score $\mathcal{I}(\mathcal{P})$ is guaranteed to be maximum. This optimal solution is unique when $\omega(e) > 0$ for all edges $e \in E$.

We model the 2-IP problem ($k = 2$) as an equivalent weighted matching problem on an undirected graph $G' = (V', E')$, which we construct as follows. We include a vertex $i \in V'$ for each edge $e_i \in E$ in the input graph. Hence $|V'| = m$. Whenever edges $\{e_i, e_j\} \in E$ form an interesting 2-path P_{ij} in G, we add the undirected edge $(i, j) \in E'$ with its weight $\omega_{ij} = \mathcal{I}(P_{ij})$ computed using Equation (1). Notice that $\omega_{ij} \geq 0$ for all edges $(i, j) \in E'$, and $|E'| \leq m(m-1)/2$. A matching $M' \subseteq E'$ in G' corresponds to a set $\mathcal{P}$ of edge-disjoint interesting 2-paths in G—a vertex $i \in V'$ will be matched with *at most* one other vertex $j \in V'$, and such a match of vertices in V' corresponds to the interesting path $P_{ij} \in \mathcal{P}$. It follows that a maximum matching in G' corresponds to an optimal solution to 2-IP on the input graph G.

The maximum weighted matching problem on an undirected graph with n vertices and m edges can be solved in strongly polynomial time—e.g., Gabow's implementation [7] of Edmonds' algorithm [6] runs in $O(nm + n^2 \log n)$ time. As such, we can solve 2-IP by solving the weighted matching problem on the associated graph G' in $O(m^3)$ time. Hence k-IP is in P for $k \leq 2$. □

We now consider k-IP for $k \geq 3$ on directed acyclic graphs. To characterize its complexity, we study the decision version of k-IP termed k-IPD, in which we are given a directed acyclic graph $G = (V, E)$ with edge weights $\omega(e) \geq 0$ and signatures $\mathrm{Sig}(e)$ for $e \in E$ and a target score $s_0 \geq 0$. The goal is to determine if there exists a collection $\mathcal{P}$ of edge-disjoint interesting k-paths in G whose total interestingness score $\mathcal{I}(\mathcal{P}) \geq s_0$.

Theorem 4.2. *k-IPD on a directed acyclic graph $G = (V, E)$ is NP-complete for $k \geq 3$.*

Proof. Given a collection $\mathcal{P}$ of interesting k-paths in a directed acyclic graph $G = (V, E)$, we can verify they are edge-disjoint, each path has k edges, and signatures are identical for all edges in each path, all in polynomial time. We can compute the interestingness score of each k-path $P \in \mathcal{P}$ using Equation (1) also in polynomial time, and add the $\mathcal{I}(P)$ values to compare with s_0 to check for equality. Hence k-IPD is in NP.

We now reduce the *exact 3-cover* problem (X3C) to 3-IPD. We then show a similar reduction for $k \geq 4$ as well, proving k-IPD is NP-complete for $k \geq 3$. The latter case for general k subsumes the case for $k = 3$. We still present the details for $k = 3$ separately, as this case reveals the structure of the general reduction in an arguably simpler setting. X3C is a version of one of the 21 NP-complete problems originally introduced by Karp [13], and is defined as follows. Given a set X with $|X| = 3q$ elements and a collection $\mathscr{C}$ of 3-element subsets of X with $|\mathscr{C}| = p$, determine if there exists a subset $\mathscr{C}' \subseteq \mathscr{C}$ such that each element of X belongs to exactly one member of $\mathscr{C}'$. Notice that such an exact cover $\mathscr{C}'$ must necessarily have exactly q members. Also, we assume $p \geq q \geq 3$ (else the instance will be trivial).

Given an instance of X3C, we create a directed acyclic graph $G = (V, E)$ for an instance of 3-IPD as follows. Each element $x \in X$ corresponds to a unique directed edge in G. Corresponding to each 3-element set $\{x, y, z\} \in \mathscr{C}$, we add to G a directed acyclic graph object as shown in Figure 3. Hence we add the 13 vertices to V and the 12 edges to E. Note that graph objects corresponding to different sets in $\mathscr{C}$ are glued together (i.e., connected) by the edges corresponding to their shared elements. The edges corresponding to all $x \in X$ are assigned the large weight $\omega = p$ making them "heavy" edges, while the rest of the edges are all assigned unit weights. Further, we assume $\mathrm{Sig}(e)$ is identical for all edges $e \in E$. The three "V"-shaped 3-paths in the top of Figure 3 are referred to as the x-, y-, and z-paths. Notice that by this construction, G can have at most $|V| \leq 13p$ vertices and $|E| \leq 12p$ edges.

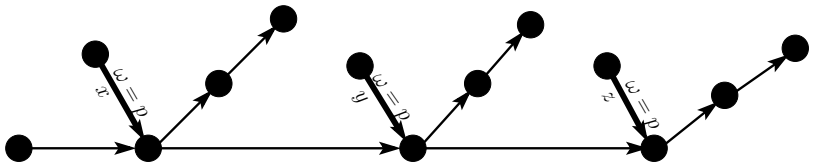
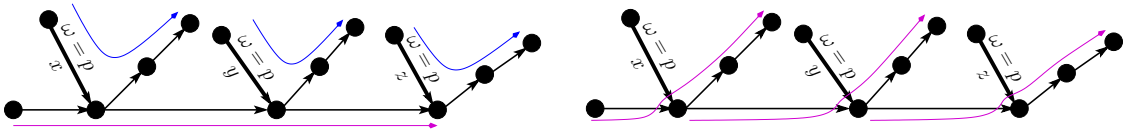


Figure 3: Graph object corresponding to set $\{x, y, z\} \in \mathscr{C}$ in X3C. Thin edges all have $\omega = 1$.

Let $W_{\text{in}} = 3(p \log 2 + \log 3 + \log 4) + (\log 2 + \log 3 + \log 4) = 4 \log 24 + 3(p-1) \log 2$, and $W_{\text{out}} = 3 \log 24$. From a graph object as shown above, we observe that 4 edge-disjoint interesting 3-paths can be chosen by 3-IPD each with interestingness score W_{in} if and only if the x-, y-, and z-paths are chosen along with *one* other interesting 3-path as shown in Figure 4a. Further, each edge corresponding to an element in X may belong to only one 3-path. Thus, at most q such graph objects may contribute the score of W_{in} to the total interestingness score. The remaining $p - q$ graph objects may contribute a score of at most W_{out} corresponding to the selection of the 3 edge-disjoint interesting 3-paths shown in Figure 4b, which avoid the edges corresponding to any $x \in X$. If q such graph objects do contribute W_{in} each to the total interestingness score, it is clear that the corresponding q triplet elements in $\mathscr{C}$ form an exact 3-cover of X. Further, 3-IPD on G will identify exactly $4q + 3(p - q) = 3p + q$ edge-disjoint interesting 3-paths with a total interestingness score of exactly $qW_{\text{in}} + (p - q)W_{\text{out}} = pW_{\text{out}} + 3(p-1)q \log 2 + q \log 24$.



(a) 3-paths for an element in $\mathscr{C}'$.

(b) 3-paths for an element not in $\mathscr{C}'$.

Figure 4: Choices of interesting 3-paths in graph objects for elements in $\mathscr{C}'$ and $\mathscr{C} \setminus \mathscr{C}'$.

Conversely, if X3C has an exact 3-cover $\mathscr{C}'$, we choose the 4 edge-disjoint interesting 3-paths (recall we assume identical signatures for all edges in G) with interestingness score W_{in} as described above in the graph object in G corresponding to each of the q 3-element sets in $\mathscr{C}'$. For the $(p - q)$ 3-element sets in $\mathscr{C} \setminus \mathscr{C}'$, we choose the 3 interesting 3-paths with

interestingness score W_{out} each in the corresponding graph objects in G. This collection of p edge-disjoint interesting 3-paths in G will have a total interestingness score of exactly $pW_{\text{out}} + 3(p-1)q\log 2 + q\log 24$.

Thus X has an exact 3-cover if and only if 3-IPD on G has a target total interestingness score of $s_0 = pW_{\text{out}} + 3(p-1)q\log 2 + q\log 24$, proving 3-IPD is NP-complete.

We now extend this result to k-IPD for $k \geq 4$. To this end, we reduce the *exact k-cover* problem (XkC) to k-IPD for general $k \geq 4$. The XkC problem is a generalization of X3C, and is defined as follows. Given a set X with $|X| = kq$ elements and a collection $\mathscr{C}$ of k-element subsets of X with $|\mathscr{C}| = p$, determine if there exists a subset $\mathscr{C}' \subseteq \mathscr{C}$ such that each element of X belongs to exactly one member of $\mathscr{C}'$. Notice that such an exact cover $\mathscr{C}'$ must necessarily have exactly q members. Also, we assume $p \geq q \geq k$ (else the instance will be trivial).

Given an instance of XkC, we create a directed acyclic graph G for an instance of k-IPD as follows. Each element $x \in X$ corresponds to a unique directed edge in G. For each k-element set $\{x_1, x_2, \ldots, x_k\} \in \mathscr{C}$, we add to G a corresponding directed acyclic graph object as shown in Figure 5. The edges corresponding to all $x \in X$ are assigned the large weight $\omega = p$ (giving "heavy" edges), while the rest of the edges are all assigned unit weights. Further, we assume $\text{Sig}(e)$ is identical for all edges $e \in E$. The k "V"-shaped k-paths in the top of Figure 5 are referred to as the x_1-, x_2-,... x_k-paths. Notice that by this construction, G can have at most $|V| \leq (k(k+1)+1)p$ vertices and $|E| \leq (k(k+1))p$ edges.

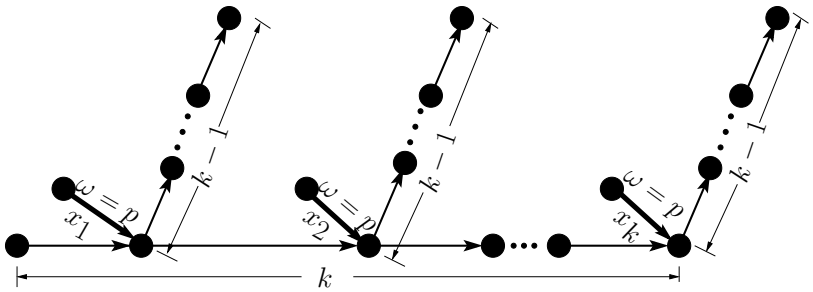
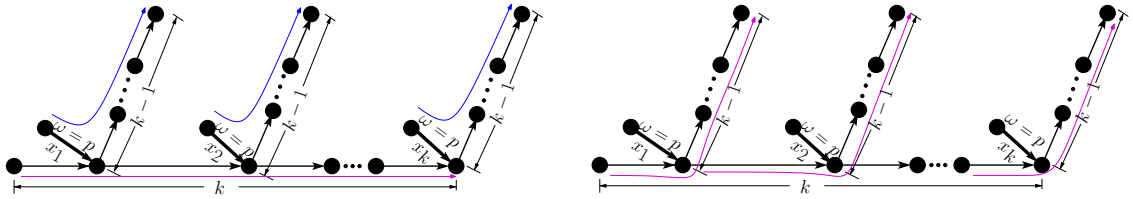


Figure 5: Graph object corresponding to set $\{x_1, x_2, \ldots, x_k\} \in \mathscr{C}$ in XkC. Thin edges all have $\omega = 1$.

Let $W_{\text{in}} = k(p\log 2 + \log 3 + \log 4 + \cdots + \log(k+1)) + (\log 2 + \log 3 + \cdots + \log(k+1)) = (k+1)\log((k+1)!) + k(p-1)\log 2$, and $W_{\text{out}} = k\log((k+1)!)$. From a graph object as shown above, we observe that $k+1$ edge-disjoint interesting k-paths can be chosen by k-IPD each with interestingness score W_{in} if and only if the x_1-, x_2-, ... x_k-paths are chosen along with *one* other k-path as shown in Figure 6a. Further, each edge corresponding to an element in X may belong to only one k-path. Thus, at most q such graph objects may contribute the score of W_{in} each to the total interestingness score. The remaining $p-q$ graph objects may contribute a score of at most W_{out} each corresponding to the selection of the k edge-disjoint interesting k-paths shown in Figure 6b, which avoid the edges corresponding to any $x \in X$. If q such graph objects do contribute W_{in} each to the total interestingness score, it is clear that the corresponding q k-tuple elements in $\mathscr{C}$ form an exact k-cover of X. Further, k-IPD on G

will identify exactly $(k+1)q+k(p-q) = kp+q$ edge-disjoint interesting k-paths with a total interestingness score of exactly $qW_{\text{in}}+(p-q)W_{\text{out}} = pW_{\text{out}}+k(p-1)q\log 2+q\log((k+1)!)$.



(a) k-paths for an element in $\mathscr{C}'$. (b) k-paths for an element not in $\mathscr{C}'$.

Figure 6: Choices of interesting k-paths in graph objects for elements in $\mathscr{C}'$ and $\mathscr{C} \setminus \mathscr{C}'$.

Conversely, if XkC has an exact k-cover $\mathscr{C}'$, we choose the $k+1$ edge-disjoint interesting k-paths (again, we assume identical signatures for all edges in G) with interestingness score W_{in} as described above in the graph object in G corresponding to each of the q k-element sets in $\mathscr{C}'$. For the $(p-q)$ k-element sets in $\mathscr{C} \setminus \mathscr{C}'$, we choose the k edge-disjoint interesting k-paths with interestingness score W_{out} each in the corresponding graph objects in G. This collection of p edge-disjoint interesting k-paths in G will have a total interestingness score of exactly $pW_{\text{out}} + k(p-1)q\log 2 + q\log((k+1)!)$.

Thus X has an exact k-cover if and only if k-IPD on G has a target total interestingness score of $s_0 = pW_{\text{out}}+k(p-1)q\log 2+q\log((k+1)!)$, proving k-IPD is NP-complete for $k \geq 4$. □

5 The Interesting Paths (IP) Problem

The goal of IP is to find a set $\mathcal{P}$ of edge-disjoint interesting paths of possibly varying lengths (1 to $n-1$) such that the sum of their interestingness scores is maximized. The determination of the precise complexity class for IP is an open problem. However, based on the hardness results for k-IP (Section 4), we conjecture that the IP is also intractable. Here we present an efficient heuristic for IP on DAGs by employing the exact algorithm for Max-IP on DAGs (in Section 3.1) as a subroutine. We also estimate lower and upper bounds on the maximum total interestingness score of IP (see Section 5.2).

5.1 An Efficient Heuristic for IP on DAGs

We present a polynomial time heuristic to find a set of edge-disjoint interesting paths $\mathcal{P}$ in a DAG with high total interestingness score. We do not provide any guarantee on the optimality or (approximation) quality of the collection of interesting paths $\mathcal{P}$.

Our method, termed Algorithm 1, uses a greedy strategy by iteratively calling the exact algorithm for Max-IP (Section 3.1). The idea is to iteratively detect a maximum interesting path, add it to the working set of solutions, remove all the edges in that path, and recompute Max-IP on the remaining graph, until there are no more edges left.

Algorithm 1: Greedy Heuristic for IP on DAGs

Input: DAG $G = (V, E)$ with $\omega(e)$, $\text{Sig}(e)$ $\forall e \in E$
Output: A set of edge-disjoint interesting paths $\mathcal{P}$ in G
$\mathcal{P} = \emptyset$
repeat
  $P \leftarrow$ Compute MAX-IP on $G = (V, E)$ and return a most interesting path
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
  Remove edges in P from E
until $E = \emptyset$;
return $\mathcal{P}$

Complexity Analysis: The runtime to compute MAX-IP on $G = (V, E)$ in the first step is $O(m\delta d_{\text{in}})$, as described in Section 3.1.1. Recall that δ is the diameter of the DAG and d_{in} is the maximum indegree of any vertex in V. Therefore, if we denote p to be the number of iterations (equivalently, the number of interesting paths found), then the overall runtime complexity is $O(pm\delta d_{\text{in}})$. However, we expect the performance of the algorithm in practice to be much faster. Note that at least one edge is, and at most m edges are, eliminated in each iteration, thereby implying $1 \leq p \leq m$ here.

Consider the worst case of elimination where one edge is eliminated in each iteration, i.e., $p = m$. The graph must be very sparse in this case, i.e., $m = \Theta(n)$, causing our algorithm for MAX-IP to perform only $O(m)$ work per iteration. Therefore the overall runtime is $O(m^2)$, or equivalently, $O(n^2)$.

On the other hand, consider the case where the number of edges reduces by a constant factor c at every iteration. This setting implies $p = O(\log_c(m))$, while the work performed from one iteration to the next will also continually reduce by a factor of c. Hence the overall runtime can still be bounded by $O(m\delta d_{\text{in}})$, the cost of MAX-IP. Further, from an application standpoint, such a greedy iterative approach can be terminated whenever an adequate number of "top" interesting paths are identified.

While the greedy heuristic promises to be efficient in practice, we show by example that it may not find the optimal solution for all instances.

Lemma 5.1. *Algorithm 1 is* not *exact for* IP.

Proof. Algorithm 1 follows a greedy strategy by iteratively calling MAX-IP. In Figure 7, we present an instance that shows this algorithm is *not* exact for IP. We assume identical signatures for all edges here.

Figure 7A) shows a weighted DAG containing two paths P_1 and P_2, followed by an edge e_8. Path P_1 consists of the edges $[e_1, e_2, e_3, e_4]$ and path P_2 consists of the edges $[e_5, e_5, e_7]$.

Algorithm 1 will output paths $\{P_2 \cup \{e_8\}, P_1\}$, because $P_2 \cup \{e_8\}$ is the most interesting path with an interestingness score of 24.5. Together with the score of 21.98 for P_1, the total interestingness score for the collection identified by the algorithm is 46.48.

However, the solution for this IP instance is $\{P_2, P_1 \cup \{e_8\}\}$, with a total interestingness score of $23.34 + 23.27 = 46.61$. □

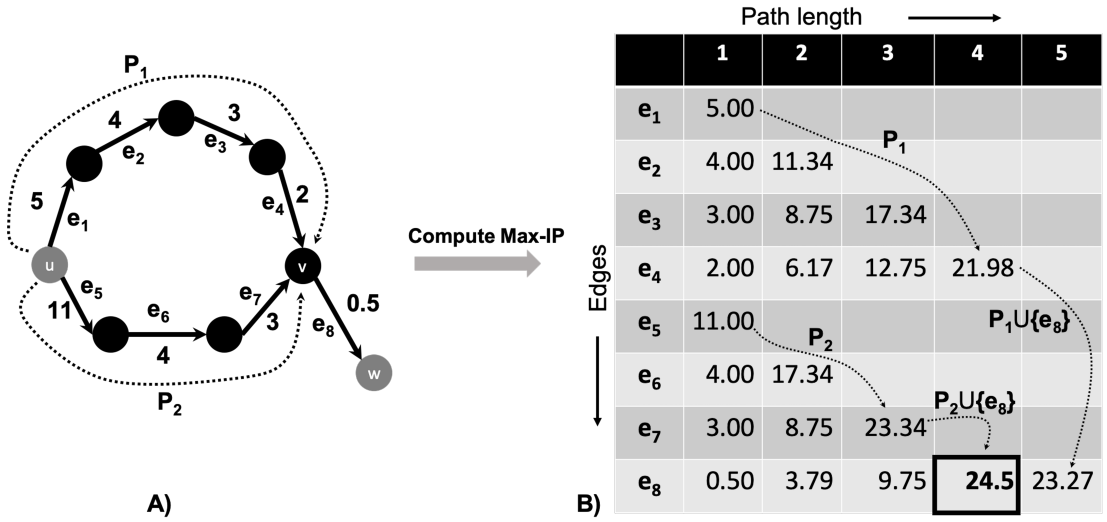


Figure 7: Counterexample to show that Algorithm 1 is not exact for IP. Part (A) shows the input DAG. Part (B) shows the tabulated computation during the first iteration of Algorithm 1.

5.1.1 An Efficient Heuristic for k-IP on DAGs

The above heuristic for IP can be easily modified to devise a heuristic for k-IP on DAGs. Algorithm 2 summarizes the main steps. The main idea is to modify the exact algorithm for MAX-IP on a DAG such that it initializes a recurrence table of size $m \times k$ (instead of $m \times (n-1)$), and then use that table to iteratively compute MAX-IP paths. The only constraint here is that each such MAX-IP path should originate from the k-th column during backtracking, so that paths output are guaranteed to be of length k. The runtime is bounded by $O(mrd_{\text{in}})$, where $r = \min\{k, \delta\}$.

Algorithm 2: Greedy Heuristic for k-IP on DAGs

Input: DAG $G = (V, E)$ with $\omega(e)$, Sig(e) $\forall e \in E$
Output: A set of edge-disjoint interesting k-paths $\mathcal{P}$ in G
Initialize an $m \times k$ table T'
$\mathcal{P} = \emptyset$
repeat
  $P' \leftarrow$ Compute MAX-IP on $G = (V, E)$ using T', and return a most interesting path *ending in column* k
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{P'\}$
  Remove edges in P' from E
until $E = \emptyset$;
return $\mathcal{P}$

Remark 5.2. Ideas used in Algorithms 1 and 2 could be combined to develop a heuristic for ATLEAST-k-IP, a modified version of k-IP that seeks to find a collection of interesting paths in G where each path has *at least* k edges. We have implemented this heuristic in our software suite (see Section 7 for details).

5.2 Bounds for IP

Let $\mathcal{P}^*$ represent an optimal set of paths for an instance of IP. We derive upper and lower bounds on its total interestingness score $\mathcal{I}(\mathcal{P}^*)$.

Let $P_{\max}(i)$ denote a maximum interesting path (of arbitrary length) ending at a given edge $e_i \in E$.

Lemma 5.3. $\mathcal{I}(\mathcal{P}^*) \leq \sum_{e_i \in E} \mathcal{I}(P_{\max}(i))$.

Proof. Consider an arbitrary path $P \in \mathcal{P}^*$. We first note that individual paths that are members of an optimal solution ($\mathcal{P}^*$) for the IP problem can end at any arbitrary *non-source* vertex in $G = (V, E)$ (see Figure 8 for an illustration).

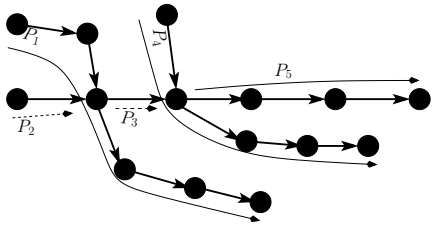


Figure 8: Five interesting paths labeled $P_1, \ldots, P_5$. The paths P_2 and P_3 end at non-source vertices.

Without loss of generality, let us assume that the input graph contains only vertices with degree at least one (as vertices with degree zero cannot contribute to any interesting path). We consider two sub-cases:

Case A: No two maximum scoring paths ending at two different edges e_i and e_j intersect, i.e., $P_{\max}(i) \cap P_{\max}(j) = \emptyset$, $\forall i, j \in [1, m]$ and $i \neq j$.
This case can occur only if the number of edges (m) is equal to the number of source vertices. This setting implies that $\mathcal{P}^*$ is comprised of m paths, where each path $P \in \mathcal{P}^*$ is a unique edge $e_i \in E$. Therefore, $\mathcal{I}(\mathcal{P}^*) = \sum_{e_i \in E} \mathcal{I}(P_{\max}(i))$ in this case.

Case B: There exists at least two maximum interesting paths ending at two different edges e_i and e_j that intersect, i.e., $P_{\max}(i) \cap P_{\max}(j) \neq \emptyset$.
This case implies that at least one of these two paths is *not* a member of $\mathcal{P}^*$ (by definition of the IP problem); let this non-member path be $P_{\max}(j)$ without loss of generality. Since all edges are covered by $\mathcal{P}^*$ by definition of IP, there still has to exist an alternative path ending in e_j that is either directly contained in $\mathcal{P}^*$ or contained as a *subpath* of a longer path in $\mathcal{P}^*$; let us refer to this alternative path as $P'(j)$. Since $P_{\max}(j)$ is an optimal interesting path ending at edge e_j, $\mathcal{I}(P'(j)) \leq \mathcal{I}(P_{\max}(j))$. In other words, the contribution of $\mathcal{P}'(j)$ to $\mathcal{I}(\mathcal{P}^*)$ cannot exceed the contribution of $P_{\max}(j)$ to $\mathcal{I}(\mathcal{P}^*)$. Therefore, $\mathcal{I}(\mathcal{P}^*) \leq \sum_{e_i \in E} \mathcal{I}(P_{\max}(i))$ in this case as well. □

We now present a lower bound for $\mathcal{I}(\mathcal{P}^*)$.

Lemma 5.4. $\mathcal{I}(\mathcal{P}^*) \geq \sum_{e_i \in E} (\omega(e_i) \times \log 2)$.

Proof. Since $\mathcal{P}^*$ covers all edges in E, a trivial (albeit not necessarily optimal) solution $\mathcal{P}'$ for IP can be constructed by including every edge as a distinct interesting path in the graph, i.e., $\mathcal{P}' = \{e_i | 1 \leq i \leq m\}$. Therefore, $\mathcal{I}(\mathcal{P}^*) \geq \mathcal{I}(\mathcal{P}') = \sum_{e_i \in E}(\omega(e_i) \times \log 2)$ follows from Equation (1). □

6 Interesting Paths in Directed Graphs

We consider a more general setting for the graph formulation (in Section 2.1.1) where the graph G need not be acyclic. This setting is motivated by the case where the weights of two nodes in G are equal, or close to equal, and hence we are not sure which way to direct the edge connecting them. Hence we add both directed edges with the same weight, as illustrated in Figure 9.

If the edge is bidirected, then the signature is used as a "wildcard"—the signature is not predetermined, and is chosen to match any candidate signature as determined by the interesting path detection algorithm. Note that in the above definition, a directed path is one where all edges in the path have the same direction. Hence we have the flexibility to use a bidirected edge as part of a directed path either in the forward or reverse direction, but not both.

We consider Max-IP on directed graphs. Unlike the case of DAGs, we show that Max-IP on directed graphs turns out to be NP-complete.

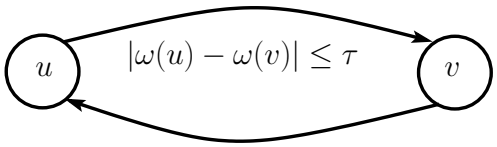


Figure 9: Two opposite directed edges between nodes u and v if $|\omega(u) - \omega(v)| \leq \tau$. Contrast this set up with the default one illustrated in Figure 1.

6.1 Max-IP on Directed Graphs

In the decision version of Max-IP termed Max-IPD, we are given a directed graph $G = (V, E)$ with edge weights $\omega(e) \geq 0$ and signatures $\text{Sig}(e)$ for $e \in E$ and a target score $s_0 \geq 0$. The goal is to determine if there exists an interesting path P in G whose interestingness score $\mathcal{I}(P) \geq s_0$.

Lemma 6.1. Max-IPD *on directed graph* $G = (V, E)$ *is NP-complete.*

Proof. Given a path P in G, we can verify that it is a directed simple path with all the edges of the same signature, compute its interestingness score $\mathcal{I}(P)$ using Equation (1), and compare it with s_0—all in polynomial time. Hence Max-IPD is in NP.

We reduce the problem of checking if a directed graph has a directed Hamiltonian cycle (DirHC) to Max-IPD. DirHC is one of the 21 NP-complete problems originally introduced by Karp [13]. Given an instance $G = (V, E)$ of DirHC with $|V| = n$, we construct an instance of Max-IPD on a directed graph $G' = (V', E')$ as follows. We replace

an arbitrary vertex $v \in V$ by two vertices v' and v'', i.e., $V' = (V \setminus \{v\}) \cup \{v', v''\}$ and $|V'| = n+1$. Each $(v, w) \in E$ is replaced by (v', w) in E' and each $(u, v) \in E$ is replaced by (u, v'') in E'. All other edges in E are included in E' without changes. All edges in E' are assigned unit weights and identical signatures, and we set $s_0 = \log((n+1)!)$.

We claim that G has a directed Hamiltonian cycle C if and only if there exists an interesting path P' in G' with interestingness score $\mathcal{I}(P') = s_0$. Let G have a directed Hamiltonian cycle C. Then C must have n edges by definition, and C visits (i.e., enters and leaves) each vertex in V exactly once. Hence there must exist edges (v, w) and (u, v) in C. We construct the interesting path P' in G' using $(v', w), (u, v'')$, and the remaining $(n-2)$ edges in C. Thus P' is a directed path in G' with n edges. Further, since all edges in E' have unit weights and identical signatures, it is clear from Equation (1) that P' is indeed an interesting path in G' with $\mathcal{I}(P') = s_0$.

Conversely, let P' be an interesting path in G' with $\mathcal{I}(P') = s_0$ (notice that $\mathcal{I}(P') > s_0$ is not possible, as nodes are not allowed to be repeated in P'). Since P is an interesting path, it visits (i.e., enters and/or leaves) any vertex in V' at most once. Since all edges in E' have unit weights and identical signatures, and by the definition of interestingness score in Equation (1), it is clear that P' must have n edges. Hence P' must start with an edge (v', w) and end with an edge (u, v''). Then the directed cycle C in G defined by the edges $(v, w), (u, v)$, and the remaining $(n-2)$ edges in P' is Hamiltonian. Hence Max-IPD on directed graphs is NP-complete. □

The result in Lemma 4.1 can be generalized to the case of directed graphs, i.e., k-IP on *directed graphs* can be solved in polynomial time for $k \leq 2$. To this end, we note a modification in the construction of the undirected graph G' for the weighted matching problem (described in the Proof of Lemma 4.1). If both interesting paths $P_{ij} = [e_i, e_j]$ and $P_{ji} = [e_j, e_i]$ are possibly formed by a pair of edges $\{e_i, e_j\} \in E$, we set $\omega_{ij} = \max\{\mathcal{I}(P_{ij}), \mathcal{I}(P_{ji})\}$.

On the other hand, since k-IP is NP-complete on DAGs for $k \geq 3$ (Theorem 4.2), it is also NP-complete on directed graphs for $k \geq 3$.

7 Software Implementation

We have implemented the algorithms for long interesting path detection as part of the `Hyppo-X` repository, which also includes our implementation of the mapper algorithm. The `Hyppo-X` repository is open source, and is available at https://xperthut.github.io/HYPPO-X/. The core computational modules are implemented in C++ and the visualization modules are implemented using the Javascript visualization library D3 [27].

The `Hyppo-X` repository includes implementations for the following algorithms presented in this paper:

1. the optimized exact algorithm for Max-IP on DAGs described in Section 3.1.1;
2. the greedy iterative heuristic for IP on DAGs described in Algorithm 1; and
3. greedy iterative heuristics for k-IP and AtLeast-k-IP on DAGs as described in Section 5.1.1.

8 Experimental Evaluation

8.1 Experimental Setup

We tested our implementations on a real-world phenomics data set obtained from an ongoing plant phenomics research project. Phenomics is an emerging branch of modern biology that involves the analysis of multiple types of data (genomic, phenotypic, and environmental) acquired using high-throughput technologies. A core goal of plant phenomics research is to understand how different crop genotypes (G) interact with their environments (E) to produce varying performance traits (phenotypes (P)); this goal is often summarized as G $\times$ E $\rightarrow$ P [16].

As a concrete study in application, we present experimental results of applying our `Hyppo-X` framework on a real-world maize data set (accessible from the `Hyppo-X` repository). This data set consists of phenotypic and environmental measurements for two maize genotypes (abbreviated here for simplicity as A and B), grown in two geographic locations (Nebraska (NE) and Kansas (KS)). These two locations capture two different growth conditions, i.e., environments. The data consists of daily measurements of growth rate alongside multiple environmental variables, over the course of the entire growing season (100 days). In our analysis, each "point" refers to a unique [genotype, location, time] combination. Consequently, the above data set consists of 400 points. Here, "time" was measured as Days After Planting (DAP), which corresponds to the number of days since planting of an individual. We tested our framework using three variables from among the environmental variables available: i) Growing Degree Days or GDD, which is a temperature-based cumulative index; ii) humidity, and iii) solar radiation.

We started our study with an intent to address a set of broad questions (hypotheses) that included the following:

i) (genotypic effect) To what degree does the genotype alone correlate with growth rate?
ii) (environmental effect) To what degree does the location (i.e., growth environment) alone correlate with growth rate?
iii) (G $\times$ E $\rightarrow$ P) How do specific genotypes respond to specific environmental variables in their performance?

In an attempt to answer these questions, we ran our `Hyppo-X` framework on the real-world maize data using different filter functions. In all our studies, we used growth rate as our performance function (i.e., dependent variable) as that is a key phenotype of interest to plant scientists. To ensure that the inferences are derived solely based on data and to prevent any pre-conceived biases, we performed all our runs in a fully unsupervised manner—i.e., the source information of the data (i.e., genotype and location) were *not* considered as input. Instead, we incorporated that information posthumously during our analysis step and used it make interpretations and inferences. Note that subpopulations of individuals that "behave" similarly under similar environmental conditions, are expected to form partial clusters, while a trail of such clusters is expected to be captured by a long interesting path.

8.2 Experimental Results

Figure 10 shows the outputs generated by running our greedy heuristic for AtLeast-k-IP on the above data set containing both A and B genotypic points. Figure 10a shows the output obtained using DAP as the filter function (i.e., independent variable), whereas Figure 10b shows the output obtained using GDD as the filter function.

Figures 11 and 12 show the output obtained using two filter functions—DAP and humidity—while considering points from genotype B (this was done to study the variance within a single genotype across the two different locations).

Figure 13 shows the output obtained using two filter functions—DAP and solar radiation—considering points from both genotypes A and B; in order to illustrate the combined effects of two variables on the entire population.

We now elaborate on each of the experimental studies.

8.2.1 Single Filter

Figure 10 shows the mapper complexes generated using single filter functions DAP (in Figure 10a) and GDD (in Figure 10b). An interesting path in this case captures a sequence of partial clusters where the average growth rates of the clusters along the path is nondecreasing. The filter function values (i.e., average DAP or average GDD values) could be either monotonically increasing or decreasing (but not both). The latter condition is guaranteed through the signature matching step in our framework.

Based on the interesting paths detected, we make the following key observations:

Observation 8.1. *The top ranked interesting paths in both cases are identified as P_1 (and shown in red in Figures 10a and 10b). In both cases, this path corresponds to the Kansas-B genotypic subpopulation, suggesting that this variety has a significantly different behavior (as per growth rate) compared to the same genotype grown at the other location (Nebraska-B; captured by path P_3). A similar distinction can be observed between Kansas-A (captured by path P_2) vs. Nebraska-A (captured by path P_4).*

Observation 8.2. *The other key observation that can be made by referring to Figure 10 is that in both locations, genotype B shows an earlier increase in growth rate (i.e., earlier by DAP* and *by GDD) compared to genotype A.*

These results show that while genotype is a likely contributor to performance (Observation 8.2), the environment (as captured by the different locations) is also a likely contributing factor (Observation 8.1).

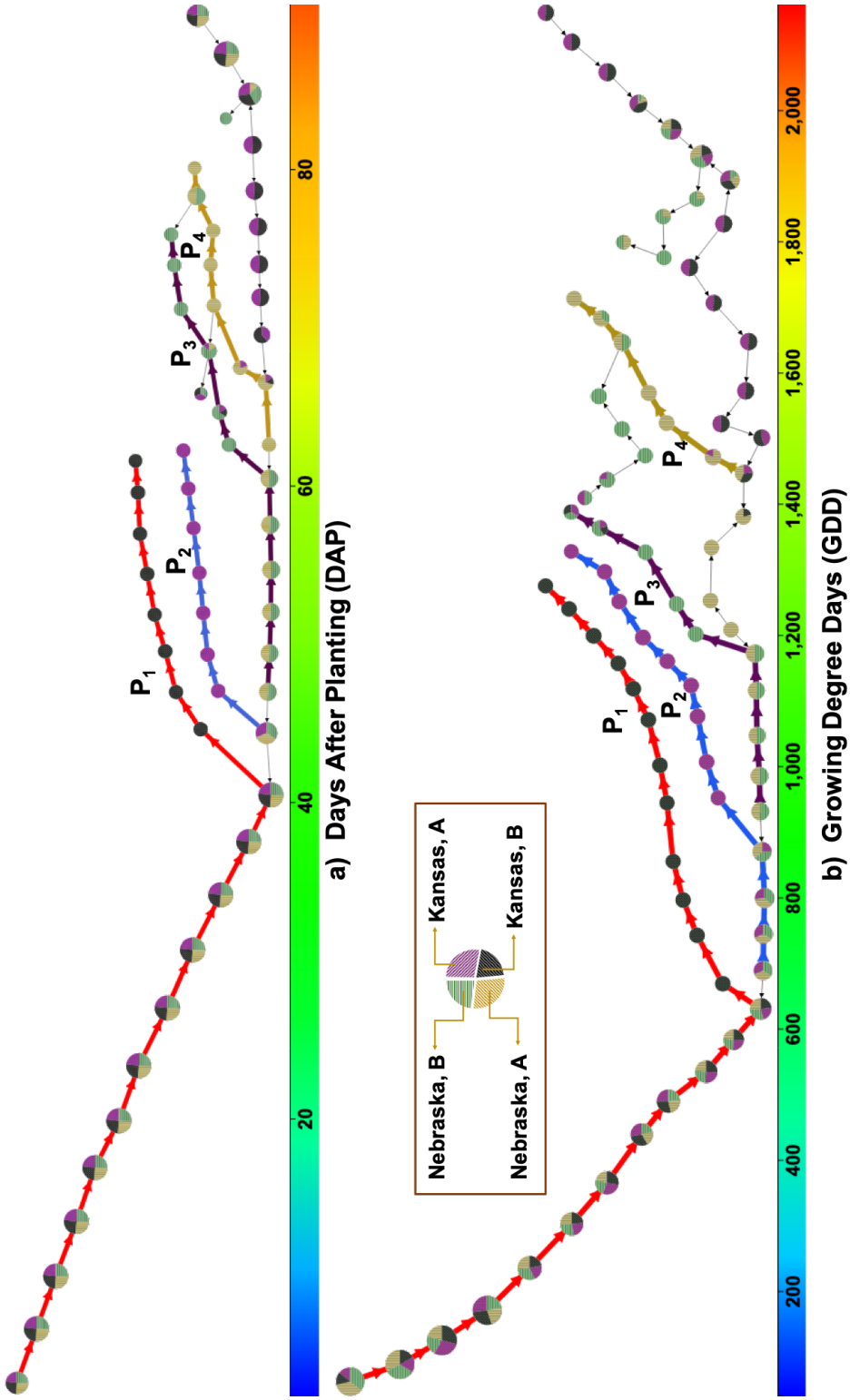


Figure 10: A screenshot of our Hyppo-X tool for identifying interesting paths in Mapper. The results are on an experimental plant phenomics data set, in which two varieties of maize (A and B) were grown in two different locations (KS and NE). Part a) shows the mapper complex constructed using DAP (Days After Planting) as the filter function, and Part b) shows the complex constructed using GDD (Growing Degree Days) as the filter function. Both complexes were constructed using growth rate as our dependent variable (i.e., target function for performance). The interesting paths are marked in different colors where each path captures a subpopulation of interest (as described in our observations).

8.2.2 Two Filters

Effect of DAP and Humidity: We study the effect of applying two filter functions to the plant data set. Specifically, we use DAP and humidity as our two filter functions and apply to the subset of points corresponding to genotype B. The choice of genotype B was motivated by its divergent growth behavior, as noted in Observation 8.2. We first show the resulting mapper complex in Figure 11. Here, each edge has two signature bits, one each for DAP and humidity, and all edges along a path have the same signature; thereby encoding the capacity to capture four different combinations of possible environmental impact on performance.

Based on the interesting paths detected (shown in Figure 11), we make the following observations:

Observation 8.3. *Paths P_1 and P_2 collectively represent the region of the complex where the Kansas-B subpopulation (as shown in Figure 11a) experiences an accelerated growth rate (as shown in Figure 11b). The paths P_3 and P_4 show the similar regions for the Nebraska-B subpopulation.*

The reason why paths P_1 and P_2 could *not* be combined to produce a longer path is because of signature matching. Specifically, the DAP increases monotonically along both paths P_1 and P_2 but humidity fluctuates, resulting in the fragmentation of the paths. In order to capture this variant behavior between the two filters, we generated an alternative mapper complex using only DAP as the signature bit (while using both DAP and humidity as filter functions).

The modified set of paths identified by this relaxed signature matching scheme is shown in Figure 12. As can be seen, the interesting paths span a longer trail of clusters, showing almost an end-to-end behavior of the KS and NE subpopulations. Notably, the separation between the paths P_1 and P_2 is mainly a result of the difference in the humidity values between the KS and NE subpopulations. This suggests an active role that humidity plays in conjunction with the genotype, on impacting growth rate.

Observation 8.4. *Also shown in Figure 12 is path P_3 that appears in the second nontrivial connected component. This region in the mapper complex corresponds to DAP values greater than 70, suggesting a lower degree of separation between the two subpopulations (KS and NE) in this later growth stage (i.e., once the crop has matured).*

Effect of DAP and Solar Radiation: Next we used DAP and solar radiation as our two filter functions, and we applied it to the entire population (containing both genotypes from both locations). Figure 13 shows the resulting mapper complex. Note that we relax the signature matching procedure to match only the DAP bit (similar to Figure 12).

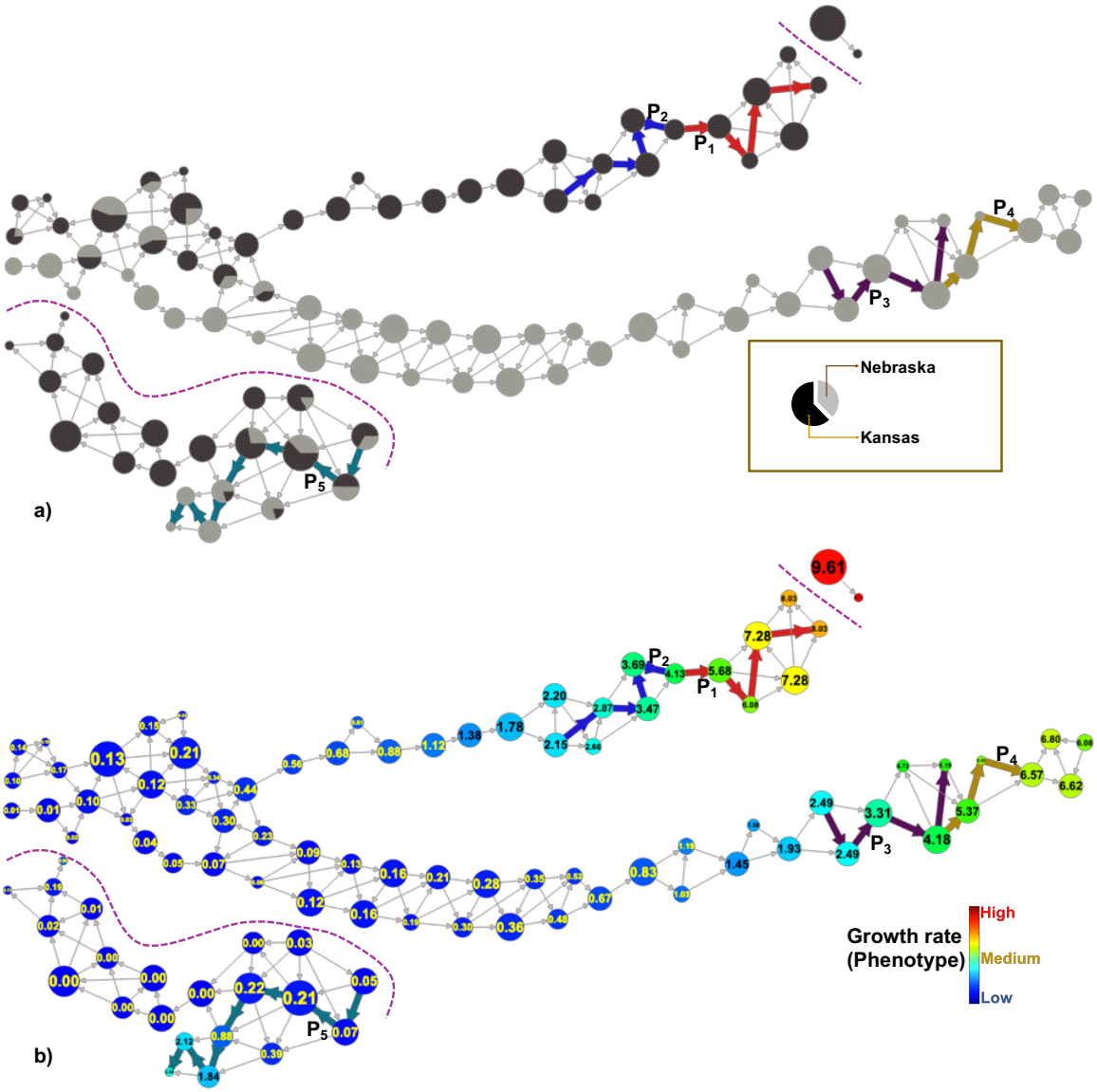


Figure 11: Two representations of the mapper complex constructed using DAP and humidity as filter functions and growth rate as the dependent variable on the subset of genotype B points. Nodes are colored based on location (a) and average growth rate (b) of points in the node. The complex contains three connected components (separated by dashed lines). We show the top interesting path(s) in each connected component.

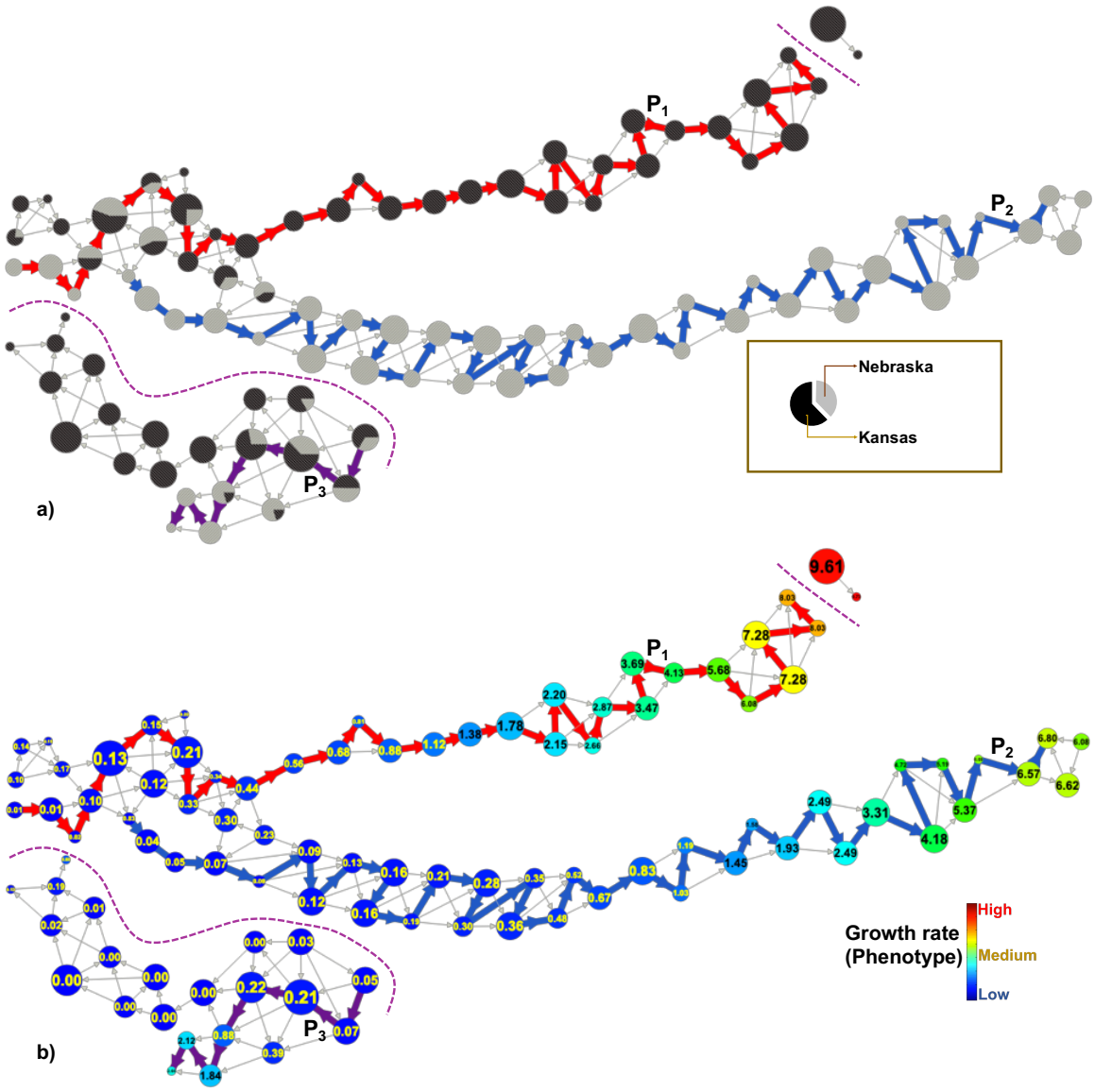


Figure 12: This mapper complex is the same as the one shown in Figure 11. Here we do not match signature for humidity because of its high variability. We obtain long interesting paths (P_1 and P_2).

Based on the interesting paths identified, we make the following key observations:

- The paths P_1, P_2 and P_5 contain subpopulations from Kansas, whereas paths P_3 and P_4 contain subpopulations from Nebraska.
- The high performing regions (see Figure 13b) within these paths correspond to genotype B.
- However, each path contains points from both genotypes along most of its length, suggesting a weak ability for solar radiation to separate the two genotypes by their performance.
- Solar radiation also does *not* produce a clear separation between the two locations (as humidity achieved)—e.g., the Kansas region P_5 is adjoining a Nebraska region (P_4).

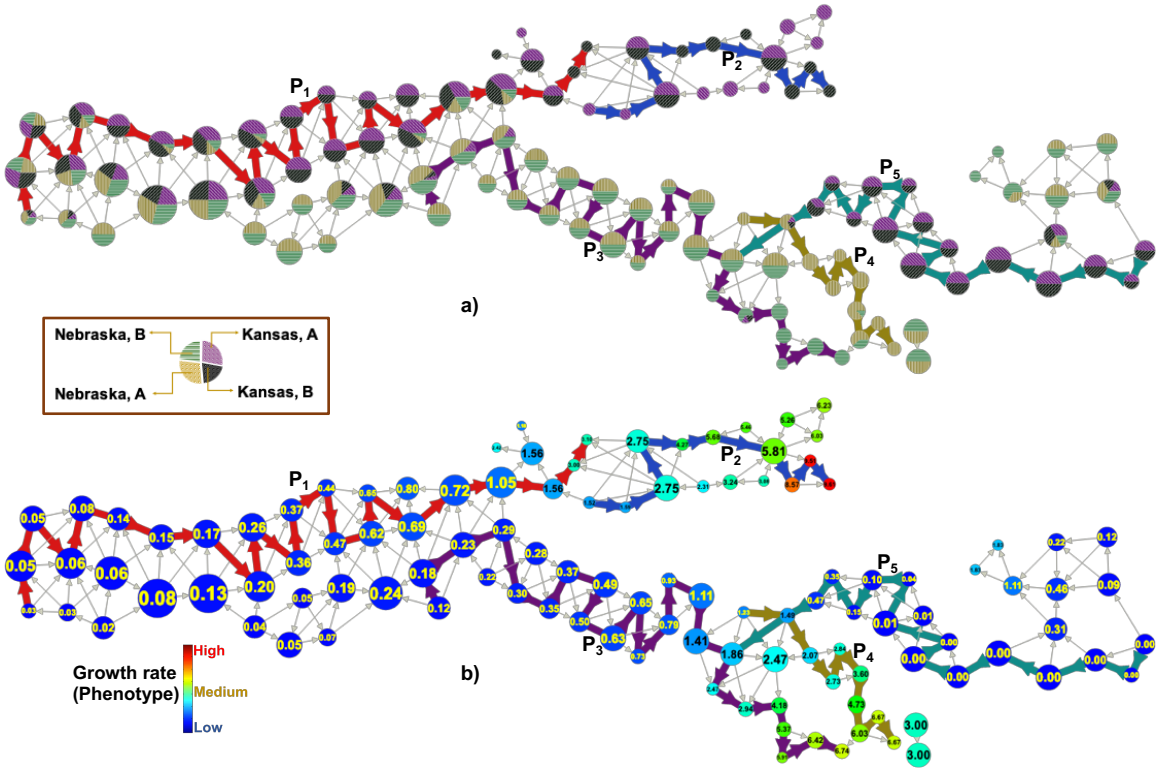


Figure 13: Two representations of mapper complex constructed using DAP and solar radiation as two filter functions and growth rate as the dependent variable on both A and B genotypic points. Nodes are colored based on location (a) and average growth rate (b) of all points in the node. We show top interesting paths obtained after relaxing the signature matching for solar radiation.

Visualizations of the outputs generated in the above analysis are available on the `Hyppo-X` repository. More details about this plant phenomics use case can be found in our related application-oriented paper [12].

9 Discussion

We have proposed a general framework for quantifying the significance of features in the mapper complex in terms of their interestingness scores. The associated optimization problems MAX-IP, k-IP, and IP constitute a new class of nonlinear longest path problems on directed graphs. We have not characterized the complexity of problem IP. Judging from the fact that k-IP is NP-complete even on DAGs, we suspect IP is NP-complete as well.

Our framework for quantifying interesting paths could be modified to quantify branches in flares as well as holes. When the graph G is a DAG, two interesting k-paths that start and end at the same pair of vertices could be characterized as a $2k$-hole, i.e., a cycle with $2k$ edges. One could alternatively use persistent homology tools to characterize holes—by identifying "long" generators around them. We have recently generalized the interestingness score of a path (in Equation (1)) to that of a flare, and studied flares as features capturing subpopulations showing divergent behavior [11]. The theoretical foundations of such flares still remain to be explored.

While we distinguished the dependent variable g used for clustering from the independent variables used as filter functions f_i for $i = 1, \ldots, h$, this distinction is not too critical for our framework. As indicated in Remark 2.5, one could use g simultaneously as a filter function along with the f_i's, and the overall analysis should still carry through. More generally, details of how to implement clustering within the mapper framework has not received much research attention. In initial work on phenomics [12], we obtained better results when using g alone to cluster within the mapper algorithm (rather than clustering using several, or even all, of the variables). It would be interesting to characterize the stability of the mapper complex to varying settings of clustering employed in its construction. For instance, could we identify a "small" subset of variables for use in clustering within the mapper framework that is optimal in a suitable sense? Another related question is how to select a small subset of filter functions from all available ones, such that the resulting mapper complex captures most, or all, of the topological information represented by the data set.

While we distinguished g from f_i, our goal in this work is not to predict g using f_i as input. Interesting paths in the mapper graph identify subpopulations that show distinct variations in g as values of f_i vary. Such insights help one develop hypotheses that the practitioner could test using further experiments [12].

The more direct question of stability of interesting paths identified by our framework is an important one, and we hope to address it in our future work. In a separate recent paper, Krishnamoorthy and coauthors have explored a notion of stability of paths in the mapper graph [2]. But the mapper graph in this work is undirected, and the edge weights capture a Jaccard distance between the clusters defining the two vertices. Stability of a most interesting path identified by MAX-IP could pose some challenges, especially on how to incorporate its optimality. The nonlinear dependence of the interestingness score on the rank of each edge in the path (the $\log(1 + \text{rank})$ factor in Equation (1)) could also be tricky to handle. Since the main motivation for this work is to identify interesting subpopulations, we might consider defining a distance function between paths that captures this information, and derive a stability result in terms of how this distance is bounded as a polynomial factor

of a small change in the input (rather than bounding the change in interestingness scores). Could we generalize such a distance distance function to the k-IP or IP cases?

A different potential approach for stability could be to incorporate our interestingness scores into the mathematical machinery recently developed to obtain results on stability and statistical convergence of the 1-D mapper complex [3, 4]. Alternatively, could we define interesting paths within a multiscale mapper framework [5], and derive stability results in that setting?

While we have proposed an efficient heuristic for IP on DAGs, we are not able to certify the quality of solution obtained by this method. On the other hand, could we devise approximation algorithms for IP or k-IP? One might have to work under some simplifying assumptions on the distribution of weights $\omega(e)$ or on the structure of the graph G. The simplest case to consider appears to that of IP on a DAG with unit weights on all edges. Another natural extension of our work on interesting paths would be to define and efficiently identify interesting *surfaces* (or generalized manifolds) in higher dimensional mapper complexes.

Acknowledgments: This research is supported by the NSF grants DBI-1661348 and DMS-1819229. Krishnamoorthy thanks Frédéric Meunier for discussion on the complexity of Max-IP while visiting MSRI.

References

[1] Muthu Alagappan. From 5 to 13: Redefining the positions in basketball. In *MIT Sloan Sports Analytics Conference*, 2012. http://www.sloansportsconference.com/content/the-13-nba-positions-using-topology-to-identify-the-different-types-of-players/.

[2] Dustin L. Arendt, Matthew Broussard, Bala Krishnamoorthy, and Nathaniel Saul. Jaccard filtration and stable paths in the Mapper. *CoRR*, abs/1906.08256, 2019. URL: http://arxiv.org/abs/1906.08256.

[3] Mathieu Carrière, Bertrand Michel, and Steve Oudot. Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research*, 19(12):1–39, 2018. arXiv:1706.00204. URL: http://jmlr.org/papers/v19/17-291.html.

[4] Mathieu Carrière and Steve Oudot. Structure and stability of the one-dimensional mapper. *Foundations of Computational Mathematics*, Oct 2017. arXiv:1511.05823. doi:10.1007/s10208-017-9370-z.

[5] Tamal K. Dey, Facundo Mémoli, and Yusu Wang. Multiscale mapper: Topological summarization via codomain covers. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 997–1013, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. arXiv:1504.03763.

[6] Jack R. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards Section B*, 69:125–130, 1965.

[7] Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 434–443, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics. arXiv:1611.07055.

[8] Geoffrey M. Guisewite and Panos M. Pardalos. Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. *Journal of Global Optimization*, 1(3):245–265, 1991. doi:10.1007/BF00119934.

[9] Timothy S.C. Hinks, Xiaoying Zhou, Karl J. Staples, Borislav D. Dimitrov, Alexander Manta, Tanya Petrossian, Pek Y. Lum, Caroline G. Smith, Jon A. Ward, Peter H. Howarth, Andrew F. Walls, Stephan D. Gadola, and Ratko Djukanović. Innate and adaptive T cells in asthmatic patients: Relationship to severity and disease mechanisms. *Journal of Allergy and Clinical Immunology*, 136(2):323–333, 2015. doi:10.1016/j.jaci.2015.01.014.

[10] David Houle, Diddahally R Govindaraju, and Stig Omholt. Phenomics: the next challenge. *Nature reviews genetics*, 11(12):855, 2010.

[11] Methun Kamruzzaman, Ananth Kalyanaraman, and Bala Krishnamoorthy. Detecting divergent subpopulations in phenomics data using interesting flares. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, pages 155–164, New York, NY, USA, 2018. ACM. doi:10.1145/3233547.3233593.

[12] Methun Kamruzzaman, Ananth Kalyanaraman, Bala Krishnamoorthy, Stefan Hey, and Patrick S. Schnable. Hyppo-X: toward a scalable exploratory framework for complex high-dimensional phenomics data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019. arXiv:1707.04362. doi:10.1109/TCBB.2019.2947500.

[13] Richard M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[14] Li Li, Wei-Yi Cheng, Benjamin S. Glicksberg, Omri Gottesman, Ronald Tamler, Rong Chen, Erwin P. Bottinger, and Joel T. Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Science Translational Medicine*, 7(311):311ra174–311ra174, 2015. doi:10.1126/scitranslmed.aaa9364.

[15] Pek Y. Lum, Gurjeet Singh, Alan Lehman, Tigran Ishkanov, Mikael. Vejdemo-Johansson, Muthi Alagappan, John G. Carlsson, and Gunnar Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3(1236), 2013. doi:10.1038/srep01236.

[16] Cathie Martin. The plant science decadal vision. *The Plant Cell*, 25(12):4773–4774, 2013.

[17] Yukio Matsumoto. *An Introduction to Morse Theory*. Europe and Central Asia Poverty Reduction and Economic Manag. American Mathematical Society, 2002.

[18] Gregory S McMaster and WW Wilhelm. Growing degree-days: one equation, two interpretations. *Agricultural and forest meteorology*, 87(4):291–300, 1997.

[19] John W. Milnor. *Morse Theory*. Annals of mathematics studies. Princeton University Press, 1963.

[20] Elizabeth Munch and Bei Wang. Convergence between categorical representations of Reeb space and mapper. *International Symposium on Computational Geometry (SOCG)*, 2016.

[21] James R. Munkres. *Elements of Algebraic Topology*. Addison–Wesley Publishing Company, Menlo Park, 1984.

[22] Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011. doi:10.1073/pnas.1102826108.

[23] Jessica L. Nielson, Jesse Paquette, Aiwen W. Liu, Cristian F. Guandique, C. Amy Tovar, Tomoo Inoue, Karen-Amanda Irvine, John C. Gensel, Jennifer Kloke, Tanya C. Petrossian, Pek Y. Lum, Gunnar E. Carlsson, Geoffrey T. Manley, Wise Young, Michael S. Beattie, Jacqueline C. Bresnahan, and Adam R. Ferguson. Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nature Communications*, 6:8581+, October 2015. doi:10.1038/ncomms9581.

[24] Matteo Rucco, Emanuela Merelli, Damir Herman, Devi Ramanan, Tanya Petrossian, Lorenzo Falsetti, Cinzia Nitti, and Aldo Salvi. Using topological data analysis for diagnosis pulmonary embolism. *Journal of Theoretical and Applied Computer Science*, 9:41–55, 2015. http://www.jtacs.org/archive/2015/1/5.

[25] Ghanashyam Sarikonda, Jeremy Pettus, Sonal Phatak, Sowbarnika Sachithanantham, Jacqueline F. Miller, Johnna D. Wesley, Eithon Cadag, Ji Chae, Lakshmi Ganesan, Ronna Mallios, Steve Edelman, Bjoern Peters, and Matthias von Herrath. CD8 T-cell reactivity to islet antigens is unique to type 1 while CD4 T-cell reactivity exists in both type 1 and type 2 diabetes. *Journal of Autoimmunity*, 50(Supplement C):77–82, 2014. doi:10.1016/j.jaut.2013.12.003.

[26] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Proceedings of the Symposium on Point Based Graphics*, pages 91–100, Prague, Czech Republic, 2007. Eurographics Association. doi:10.2312/SPBG/SPBG07/091-100.

[27] Swizec Teller. *Data Visualization with d3.js*. Packt Publishing Ltd, 2013.

[28] Brenda Y. Torres, Jose Henrique M. Oliveira, Ann Thomas Tate, Poonam Rath, Katherine Cumnock, and David S. Schneider. Tracking resilience to infections by mapping disease space. *PLoS Biol*, 14(4):1–19, 04 2016. doi:10.1371/journal.pbio.1002436.

[29] George Tsaggouris and Christos Zaroliagis. *Non-additive Shortest Paths*, pages 822–834. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-30140-0_72.

[30] Loring W. Tu. *An Introduction to Manifolds*. Universitext. Springer New York, 2nd edition, 2011.

[31] Hoang Tuy, Saied Ghannadan, Athanasios Migdalas, and Peter Värbrand. The minimum concave cost network flow problem with fixed numbers of sources and nonlinear arc costs. *Journal of Global Optimization*, 6(2):135–151, Mar 1995. doi:10.1007/BF01096764.